TRABALHO DE GRADUAÇÃO

Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados

Gabriel Luiz Pedreira Cataldi

Brasília, dezembro de 2017

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados

Gabriel Luiz Pedreira Cataldi

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Geovany Araújo Borges, ENE/UnB _____
Orientador

Prof. Henrique Cezar Ferreira, ENE/UnB _____
Examinador interno

Prof. Renato Alves Borges, ENE/UnB _____
Examinador interno

Brasília, dezembro de 2017

FICHA CATALOGRÁFICA

CATALDI, GABRIEL LUIZ PEDREIRA
Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados

[Distrito Federal] 2017.

x, 99p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2017). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Birrotor 2.VANT
3. Robótica 4.Controle

I. Mecatrônica/FT/UnB II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

CATALDI, GABRIEL LUIZ PEDREIRA, (2017). Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n°018, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 99p.

CESSÃO DE DIREITOS

AUTOR: Gabriel Luiz Pedreira Cataldi

TÍTULO DO TRABALHO DE GRADUAÇÃO: Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados.

GRAU: Engenheiro ANO: 2017

Gabriel Luiz Pedreira Cataldi

Campus Darcy Ribeiro, SG-11, Universidade de Brasília

70919-970 Brasília – DF – Brasil.

Dedicatória

Dedico este trabalho aos meus pais, Cesar Cataldi e Rosangela Cataldi.

Gabriel Luiz Pedreira Cataldi

Agradecimentos

Inicialmente gostaria de agradecer à Deus e aos meus pais, Cesar Cataldi e Rosangela Cataldi, sem os quais nada seria possível. Vocês são todos os dias um exemplo e uma inspiração, que me guiam e me fazem nunca desistir dos meus sonhos. Agradeço aos meus irmãos Pedro Cataldi e Safira Cataldi, que me ajudam com cada palavra, cada dica e cada gesto. Agradeço a minha namorada Raphaella Candaten, por estar sempre ao meu lado nos momentos mais difíceis e por sempre arrumar alguma forma de me alegrar e me trazer força. Agradeço também pelas revisões realizadas com tanto carinho pela Safira, pela Raphaella e pela Rosangela. Vocês contribuíram muito para que esse texto chegasse ao fim da melhor forma possível.

Agradeço ao professor Geovany Borges por ter aceitado me orientar durante a realização deste projeto. Seu conhecimento e sua sabedoria foram essenciais para que o trabalho fosse executado.

Agradeço a todos os meus amigos da vigésima sétima turma de engenharia mecatrônica da Universidade de Brasília. O companheirismo de cada um de vocês todos os dias foi essencial para superar cada uma das batalhas travadas, e não foram poucas. Todos, sem exceção, contribuíram para a realização deste trabalho. Destaques especiais neste último ano vão para meu amigo Matheus Takahashi, com quem aprendi a fazer circuitos eletrônicos enquanto trabalhávamos no Laboratório de Engenharia Biomédica e quem estava sempre disposto a me ajudar, para Rafael Lima, quem me ajudou quando eu estava perdido no meio do projeto do software, e para Vitor Rezende, me ajudando e lembrando dos prazos. Agradeço também a toda equipe do LARA pelo aprendizado de cada dia através dos projetos, agradeço especialmente Miguel e Breno que estiveram a todo o tempo disponíveis a me ajudar na fabricação das placas de circuito.

E por fim, agradeço aos meus amigos de infância Daniel Martins e Vitor Coimbra pelo apoio, especialmente durante esse ano.

Gabriel Luiz Pedreira Cataldi

RESUMO

Um veículo aéreo *tilt rotor* é uma aeronave que consegue realizar voos estáveis de baixa velocidade, decolar e pousar na vertical como helicópteros, e atingir velocidades maiores de forma semelhante a um avião. Por isso é uma aeronave híbrida com muitas possibilidades de pesquisa, visto que apresenta uma adaptabilidade para diferentes situações. A proposta deste trabalho consiste: (i) na avaliação e no reajuste de todo o *hardware* e *software* do sistema robótico aéreo não tripulado (VANT) com dois rotores articulados do Laboratório de Robótica Aérea da Universidade de Brasília; (ii) e na implementação de um sistema de controle e estabilização para a aeronave com foco em seu controle de atitude e de orientação, com o objetivo de atingir um voo pairado estável. Para isso, uma abordagem direta foi utilizada baseada em controladores Proporcional (P) e Proporcional-Derivativo (PD), e um novo protótipo com um sistema eletrônico embarcado foi construído. Dois modos de controle foram implementados. O primeiro, utiliza um controlador P que atua no controle do veículo através da utilização de um joystick pelo usuário. O segundo, utiliza um controlador PD para realizar o controle da dinâmica de orientação do veículo nos três eixos de movimento: na guinada, na arfagem e na rolagem. Os resultados obtidos mostram o funcionamento de ambos os modos, levando a aeronave a um novo estágio de desenvolvimento capaz de voar.

Palavras-Chave: VANT; aeronave; birrotor; controle; robótica; atitude; estabilização.

ABSTRACT

A tilt rotor aerial vehicle is an aircraft with the ability to reach stable flights at low speed, to perform vertical take-off and landing like a helicopter, and to reach high speed flight as a fixed wing airplane. That is why is a hybrid aircraft with many research possibilities, since it shows great adaptability to different situations. The purpose of this work is: (i) to evaluate and readjust all of the hardware and software of the unmanned aerial robotic system with two articulated rotors (UAV) from Aerial Robotics Laboratory of University of Brasília; (ii) and to implement a control and stabilization system for the aircraft focusing on its attitude and orientation control, aiming at achieving a stable hover flight. In order to do that, it was used a direct approach based on Proportional (P) and Proportional-Derivative (PD) controllers, and a new prototype with an electronic embedded system was built. Two control modes were implemented. The first one, uses a P controller that promotes the interface between the user and the joystick. And the second, uses a PD controller to control the vehicle orientation dynamics in the three axes of movement: yaw, pitch and roll. The results show that both modes are working and lead the aircraft to a new development stage capable of flying.

Keywords: UAV; aircraft; bi-rotor; control; robotics; attitude; estabilization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

V	Tensão	[V]
I	Corrente	[A]
R	Resistência	[Ω]
P	Potência	[W]
C	Capacitância	[F]
h	Altura	[cm]
t	Tempo	[s]
v_s	Velocidade do Som	[m/s]
e	Erro do sistema de controle	
u	Ação de controle	
K_P	Ganho Proporcional	
K_D	Gaho Derivativo	
K_I	Gaho Integral	
Sd	Posição angular do Servo Direito	[º]
Se	Posição angular do Servo Esquerdo	[º]
Pd	Largura do Pulso PWM que indica a Potência do Motor Direito	
Pe	Largura do Pulso PWM que indica a Potência do Motor Esquerdo	

Símbolos Gregos

τ	Constante de Tempo	[s]
θ	Rotação no eixo x - Arfagem	[º]
ϕ	Rotação no eixo y - Rolagem	[º]
ψ	Rotação no eixo z - Guinada	[º]
β	Variação angular nos servomotores	[º]

Grupos Adimensionais

X, Y, Z, A	Eixos do *Joystick*
x, y, z	Eixos do sistema de coordenadas fixo no centro de massa do veículo aéreo

Subscritos

ref	referência
des	desejado
ADC	conversor análogico/digital

Sobrescritos

$\cdot$	Derivada com relação ao tempo

Siglas

API	*Application Programming Interface* - Interface de Programação de Aplicações
ABS	*AcrilonitrilaButadieno Estireno*
VANT	Veículo Aéreo Não Tripulado
VTOL	*Vertical Take-off and Landing*
LED	*Light-Emitting Diode*
SDA	*Serial Data Line*
SCL	*Serial Clock Line*
CS	Seletor de *Chip* - *Chip Select*
MOSI	*Master-Out/Slave-In*
MISO	*Master-In/Slave-Out*
SCK	*Serial Clock*
PCB	*Printed Circuit Board* - Placa de Circuito Impresso
CNC	*Computer Numeric Control* - Controle Numérico Computadorizado
IMU	*Inertial Measurement Unit* - Unidade de Medição Inercial
ROS	*Robot Operating System*
USB	*Universal Serial Bus*
SPI	*Serial Peripheral Interface*
GPS	*Global Positioning System* - Sistema de Posicionamento Global
SD	*Secure Digital* - Cartão SD
ESC	*Electronic Speed Control*
PWM	*Pulse-Width Modulation*
A/D	Analógico/Digital
I/O	*Input/Output* - Entrada/Saída
CPU	Unidade Central de Processamento - *Central Processing Unit*
CAD	*Computer Aided Design* - Desenho Assistido por Computador
IDE	*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
RHR	*Right Hand Rotation*
LHR	*Left Hand Rotation*
UnB	Universidade de Brasília
LARA	Laboratório de Robótica e Automação

Capítulo 1

Introdução

"É no ínicio que se deve tomar, com máxima delicadeza, o cuidado de dar às coisas sua devida proporção" - Frank Herbert (Duna)

1.1 Contextualização

Um dos sonhos do ser humano sempre foi voar, e desde muito tempo atrás, invenções foram construídas como tentativas para ganhar o céu. Leonardo da Vinci (1452-1519) foi o primeiro a ser creditado com um projeto de desenvolvimento de algo parecido com um helicóptero, veículo concebido para levantar do chão verticalmente. Baseada em uma forma de parafuso helicoidal, sua composição estrutural era formada por madeira e arames, como pode ser visualizado em seu desenho na Figura 1.1[1]. Infelizmente era só um conceito; não existia tecnologia suficiente na época para colocar sua ideia em prática. Cerca de 4 séculos depois, veio a construção de um primeiro veículo aéreo mais pesado do que o ar que realmente voasse, ou melhor, dois veículos de uma só vez, um desenvolvido pelos irmãos Wright, que voou em 1903, e outro desenvolvido pelo brasileiro Alberto Santos Dummont, que voou em 1906. Já o primeiro helicóptero a obter sucesso em um voo demonstrando performance e controle de precisão veio apenas 30 anos depois, em 1937, com o modelo Focke-Wulf Fw-61; isso em uma época em que os aviões já estavam em plena utilização.

Diante de aviões e helicópteros, surgiu então um novo desafio: juntar as funcionalidades desses dois em uma única aeronave. Apesar de ambos estarem bastante difundidos em numerosas aplicações civis e militares, era necessária a criação de um veículo que pudesse agrupar a capacidade de decolagem e pouso na vertical de um helicóptero (VTOL, do inglês, *Vertical Take-off and Landing*), com a capacidade de alcance e velocidade de um avião. Foi assim que, nas décadas de 1920 e 1930, surgiram diversas propostas tentando achar uma solução para essa questão, uma que abrangesse tanto voo em modo pairado quanto voo em modo cruzeiro, cumprindo com critérios econômicos e operacionais ao mesmo tempo. Daí surgiram os primeiros veículos aéreos birrotores articulados ou vetorizados, conhecidos em inglês como *tilt rotors*, chamados assim por possuírem rotores propulsores presos em mecanismos rotatórios, localizados nas duas extremidades de uma asa fixa. Alguns outros tipos de birrotores também foram apresentados na época, como os *tilt wings* que não possuem asa fixa. Estes, entretanto, não serão discutidos por estarem fora do

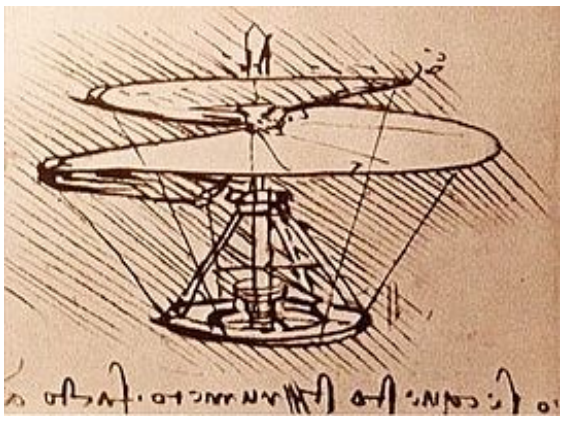


Figura 1.1: Projeto de uma máquina voadora de Leonardo da Vinci no século XV (Fonte: foto retirada de [1]).

escopo deste trabalho.

A Figura 1.2 apresenta dois conceitos, o da esquerda foi um dos primeiros na categoria *tilt rotor*, projetado e construído por Henry Berliner em 1924, e o da direita foi o primeiro a ser patenteado nos Estados Unidos em 1930, por George Lehberg[1].

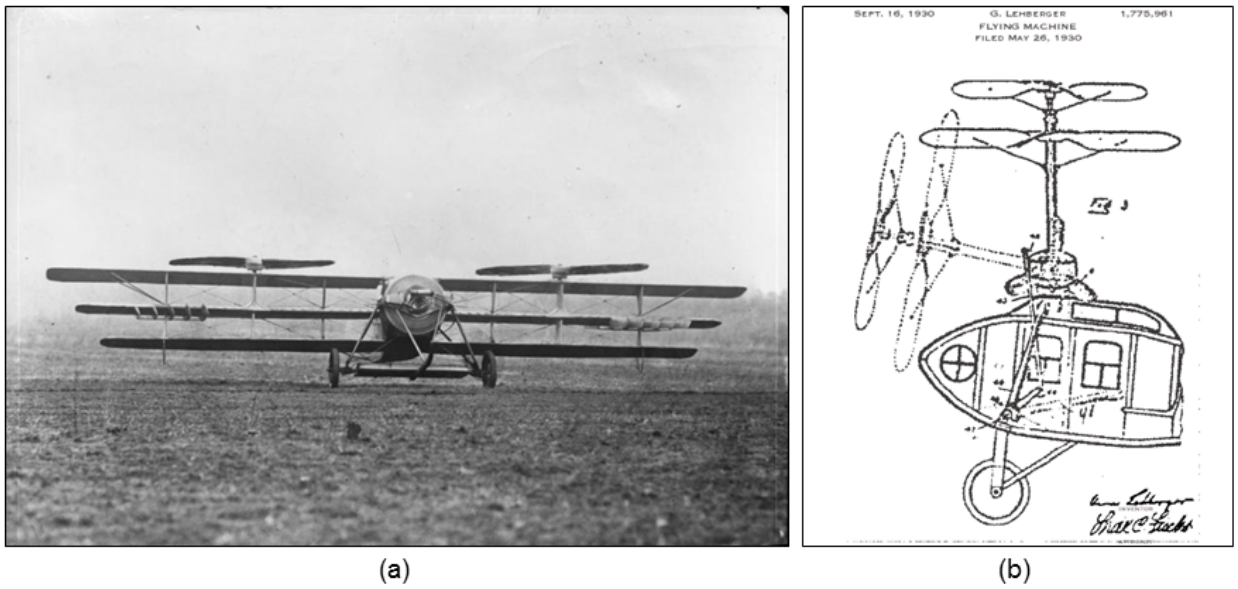


Figura 1.2: Modelos antigos de aeronaves da categoria *tilt rotor* (Fonte: foto (a) retirada do *National Air and Space Museum–NASM* e foto (b) retirada de [1]).

Atualmente, existem muitos birrotores da categoria *tilt rotor* sendo desenvolvidos. Apesar de serem vistos em sua maioria em aplicações militares, principalmente nas que exigem pouso e decolagem em terrenos irregulares de difícil acesso e nas que precisam de velocidades e alcances maiores, existem modelos recentes com propostas mais modernas sendo trabalhados, como é o caso do *AgustaWestland Project Zero*, apresentado na Figura 1.3[1], totalmente elétrico e com um

[1]Fonte: <http://www.leonardocompany.com>

visual mais voltado para utilização comercial.

Figura 1.3: Modelo recente de birrotor desenvolvido pela empresa *AgustaWestland*.

1.2 Veículos Aéreos Não Tripulados

É nessa linha de invenções e inovações que surgiram os primeiros veículos aéreos não tripulados, ou VANTs, conhecidos em inglês como UAV (*Unmanned Aerial Vehicle*), que por definição apresentada na publicação ICA 100-40, de março de 2017, do Ministério da Defesa, são todos e quaisquer tipos de aeronaves que não precisam de um piloto ou que podem ser controladas remotamente[2]. Podem ser classificadas ainda como automáticas, caso possibilitem a intervenção do piloto a qualquer momento, seja na condução ou no gerenciamento de voo, ou autônomas, caso não possua qualquer interferência do piloto. A ideia da criação de veículos aéreos não tripulados surgiu com a possibilidade de retirar o fator humano e reduzir para zero os riscos em um elevado número de aplicações, principalmente militares, como atividades de avaliação do território inimigo ou até mesmo em manobras ofensivas. Com o passar do tempo e com a difusão da tecnologia, tornando-a mais acessível para a população em geral, o desenvolvimento de veículos aéreos começaram a aparecer para os mais diferentes propósitos. Na atualidade é possível encontrá-los na agricultura monitorando plantações, em gravações de filmes, em inspeções de prédios e até mesmo em serviços de entrega, como o oferecido pela empresa *Amazon*, o *Amazon Prime Air Delivery*[3].

O termo "veículo aéreo não tripulado" não se refere somente à aeronave, mas também ao conjunto de componentes e equipamentos que esse complexo sistema engloba. Segundo os autores do livro *Small Unmanned Aircraft*[6], a definição de VANT inclui tudo que compõe a arquitetura de seu complexo sistema, seus sensores, os microcontroladores, o *software*, as estações de controle, a interface com o usuário, os atuadores e as comunicações em nível de *hardware*. De maneira simplificada, a Figura 1.4 ilustra os principais elementos da arquitetura de um VANT. A unidade

[2]Definição retirada de: "Sistemas de Aeronaves Remotamente Pilotadas e o Acesso ao Espaço Aéreo Brasileiro". Publicação ICA 100-40 do Ministério da Defesa, Comando da Aeronáutica. Disponível em: <https://publicacoes.decea.gov.br/?i=publicacao&id=4510>. Acesso em: 21 de novembro de 2017.

[3]*Amazon Prime Air Delivery*, disponível em: <https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011>. Acesso em: 21 de novembro de 2017.

de processamento é o cérebro que comanda todas as outras operações, responsável por interpretar os dados aferidos pelos sensores, calcular o controle e encaminhar as ordens até os atuadores, que no caso de um veículo áereo são seus motores. Tudo isso necessita estar conectado a uma fonte de energia, leve e com capacidade suficiente para garantir uma autonomia de voo suficiente para a tarefa para a qual foi projetada. Para monitoramento e aquisição de dados, geralmente, existe uma estação de controle, composta por um computador que não esteja embarcado na aeronave.

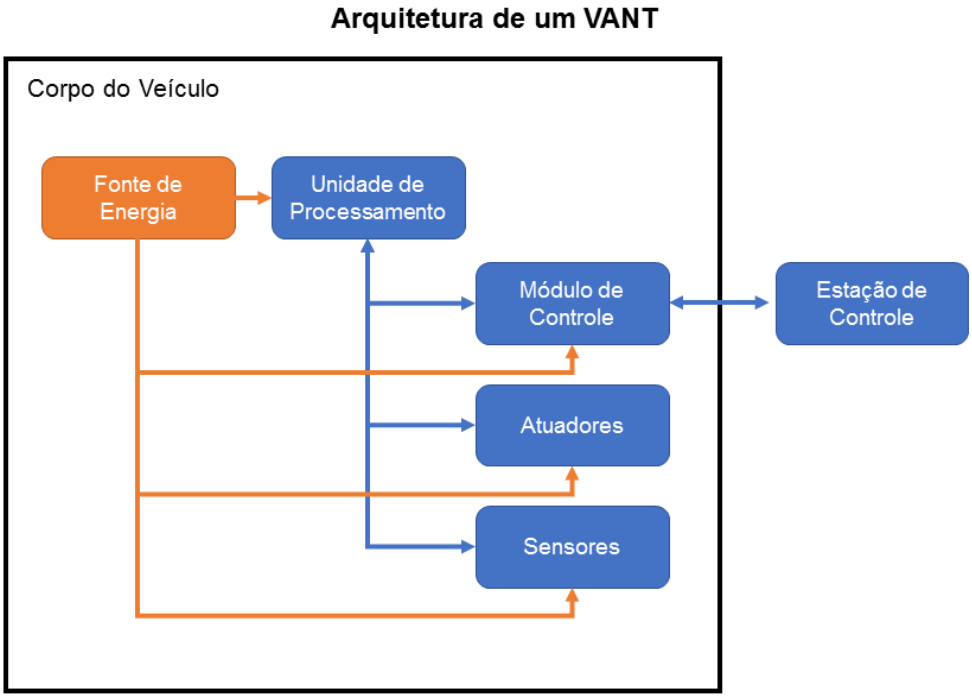


Figura 1.4: Arquitetura simplificada de um VANT (Adaptado de [2]).

Diante de tudo que foi exposto, surgiu a proposta de construir um veículo birrotor articulado não tripulado do tipo *tilt rotor* no Laboratório de Robótica Aérea (AeroLab) da Universidade de Brasília. Trabalho este iniciado por Davi Feques Vale e Mickael Ângelo A. da Costa como projeto de conclusão do curso de Engenharia de Controle e Automação, que originou o primeiro modelo de VANT VTOL na categoria *tilt rotor*[5] desenvolvido na UnB. Por meio da orientação do professor Geovany Araújo Borges, surgiu a oportunidade de dar prosseguimento a esse projeto desenvolvido no ano de 2016, e neste documento aqui apresentado será relatada, no mesmo contexto, sua continuação.

1.3 Descrição do problema e motivação

Um veículo aéreo *tilt rotor* é uma aeronave híbrida, que consegue não apenas realizar voos estáveis em baixas velocidade, decolar e pousar na vertical igual a um helicóptero, como também atingir velocidades maiores semelhante à aeronaves de asa fixa. Esse tipo de veículo possui bastante futuro, principalmente por apresentar uma incrível adaptabilidade de operação em diferentes tipos de ambientes devido as suas habilidades de voo. Por ser uma abordagem diferente, ainda existem

muitos problemas, como a interferência aerodinâmica entre os rotores e a complexa dinâmica dos mecanismos de voo. Apesar disso, aeronaves desse tipo já foram desenvolvidas com sucesso e colocadas em uso, como é o caso do modelo militar multifunção *V-22 Osprey*[4]. Isso inspira que mais trabalhos e pesquisas sejam realizados no mesmo contexto. Diante das vantagens e dos desafios apresentados, a criação de um VANT com as mesmas características se torna um trabalho motivador, o que se pode ver pelas diferentes pesquisas sendo feitas nessa área ao redor do mundo, como é o caso dos trabalhos apresentados em [7], [8] e [9].

1.4 Objetivos do projeto

A proposta deste trabalho consiste na avaliação e no reajuste de todo o *hardware* e *software* do sistema robótico aéreo com dois rotores articulados apresentado em [5], e na implementação de um sistema de controle e estabilização para a aeronave com foco em seu controle de atitude e de orientação. Para atingir essas finalidades, os seguintes objetivos específicos foram estabelecidos:

1. Avaliar e estudar todo o *hardware* do sistema, desde sua estrutura mecânica até seus circuitos;
2. Definir a topologia do sistema, identificando seus componentes e sua arquitetura;
3. Realizar o projeto eletrônico completo, apresentando os esquemáticos dos circuitos, os projetos das placas de circuito impresso e executar a sua fabricação;
4. Identificar mudanças necessárias no projeto mecânico e construir um novo protótipo, caso necessário;
5. Remodelar e reprojetar o *software*, de modo a gerenciar todo o funcionamento do sistema embarcado do VANT em uma aplicação em tempo real;
6. Desenvolver uma interface de controle entre o veículo e um *joystick*;
7. Implementar um algoritmo de controle e de estabilização de atitude e de orientação do veículo aéreo;
8. Testar e avaliar os resultados obtidos.

A partir disso, espera-se que os conhecimentos adquiridos durante o curso de Engenharia de Controle e Automação na UnB sejam aplicados, e que o resultado desse trabalho possa contribuir, dar continuidade e ser utilizado em futuras pesquisas no Laboratório de Robótica Aérea (LRA).

[4] *V-22 Osprey* da empresa *Bell Boeing*. Disponível em: <http://www.bellhelicopter.com/military/bell-boeing-v-22>. Acesso em 22 de novembro de 2017.

1.5 Resultados obtidos

O trabalho desenvolvido resultou em um protótipo atualizado de um robô aéreo birrotor, apresentando um novo projeto mecânico, eletrônico e de *software*. Agora, a plataforma apresentada possui um sistema embarcado capaz de funcionar em tempo real, com novos componentes adquiridos, com todos os módulos de sensoreamento funcionando e com uma fonte de energia própria capaz de dar autonomia de voo suficiente. Além disso, foi implementado um controlador proporcional que realiza a interface entre os movimentos do veículo e um *joystick*, e foi implementado também um algoritmo de controle de atitude e orientação utilizando um controlador proporcional-derivativo.

1.6 Apresentação do manuscrito

Esse documento está dividido em cinco capítulos. O Capítulo 2 apresenta toda a fundamentação teórica necessária para a compreensão dos elementos abordados durante o texto. No Capítulo 3 são expostas as etapas de pesquisa e desenvolvimento, dividas em duas seções mais importantes, o projeto de *Hardware* e de *Software*, apresentados na ordem de execução dos procedimentos ao longo da pesquisa. O Capítulo 4 apresenta os resultados e uma análise de cada um deles. Por fim, o Capítulo 5 traz as conclusões e também as perpectivas futuras de continuação do trabalho. Em anexo, encontram-se o diagrama eletrônico do módulo de alta corrente, o desenho técnico do projeto mecânico, uma descrição do CD que contém a versão digital do trabalho.

Capítulo 2

Fundamentos

Este capítulo apresenta uma revisão dos conceitos mais importantes que fundamentaram teoricamente o desenvolvimento do veículo aéreo birrotor.

2.1 Representações Angulares

Neste trabalho foi usado um sistema de orientação baseado em uma representação angular. Em aplicações aéreas é bastante utilizada a convenção RPY, que vem do inglês *Roll-Pitch-Yaw*, que significa rolagem, arfagem e guinada, respectivamente. Cada um corresponde ao movimento de rotação em torno de um eixo específico. De acordo com o posicionamento do sensor de medição inercial no veículo birrotor desenvolvido, usaremos as seguintes convenções:

- **Guinada**: rotação em torno do eixo z definido, indicado pelo ângulo ψ;
- **Arfagem**: rotação em torno do eixo x definido, indicado pelo ângulo θ;
- **Rolagem**: rotação em torno do eixo y definido, indicado pelo ângulo ϕ.

A Figura 2.1 mostra a representação angular e o sistema de orientação adotados durante a realização deste trabalho. Para representar o sistema de coordenadas fixo no corpo da aeronave serão utilizadas letras minúsculas mostrando os eixos x, y e z. Letras maiúsculas serão usadas para representar o sistema de coordenadas no *joystick*, portanto com os eixos retratados na forma X, Y e Z.

2.2 Sistemas Embarcados

Sistemas embarcados são definidos em [10] como sistemas computacionais microprocessados, completamente encapsulados em um dispositivo, em uma integração de *hardware* e *software* projetada para executar um conjunto de funções específicas. A palavra embarcada reflete o fato de que são realmente imbutidos em um conjunto maior. Atualmente, sistemas embarcados estão

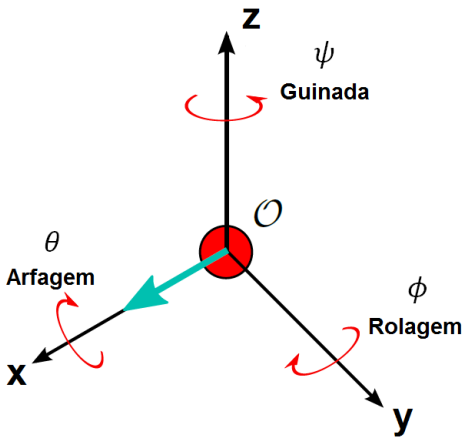


Figura 2.1: Representação do sistema de orientação RPY (Adaptado de [2]).

presentes em quase tudo que nos rodeia, seja em um automóvel, nos *smartphones* ou então em um aparelho de microondas, tornando nosso dia-a-dia cada vez mais fácil.

Por serem um tipo de sistema computacional, a constituição do seu *hardware* pode ser baseada em cima do modelo de Von Newmann, resultado de uma publicação de 1945, que definiu os requisitos de um computador eletrônico geral[11]. Em um nível mais alto, os cinco mais importantes componentes em um sistema embarcado são:

- **CPU**: do inglês *Central Processing Unit*. É o processador principal, responsável por processar instruções e dados;
- **Memória**: onde o *software* do sistema é armazenado;
- **Dispositivos de Entrada**: processadores auxiliares de entrada e componentes eletrônicos;
- **Dispositivos de Saída**: processadores auxiliares de saída e componentes eletrônicos;
- **Barramentos e caminhos de dados**: interconectam os componentes, fornecendo um caminho para que haja troca de informações entre um e outro.

Sistemas embarcados são construídos em sua quase totalidade utilizando microcontroladores (μC), definidos em [12] como pequenos computadores em um pequeno circuito integrado formado por uma simples CPU combinada com dispositivos periféricos, como memórias, temporizadores e dispositivos de entrada e saída (I/O).

Em sistemas embarcados, o cálculo de potência de seus elementos é muito importante, porque cada um exige uma quantidade mínima diferente e consegue lidar apenas com um limite máximo fornecido. É necessário saber os requisitos de potência total consumida para escolher o melhor modo de alimentação para o projeto eletrônico, de modo a garantir a quantidade suficiente de energia para um bom funcionamento do sistema.

A utilização de capacitores também é muito importante. Os de valores pequenos são geralmente usados para desacoplamento, comumente chamados de capacitores *Bypass*. Sua função é suprimir

ruídos de altas frequências em sinais de alimentação de energia, retirando pequenas ondulações de tensão que podem ser perigosas em circuitos integrados delicados. De certa forma, eles acabam atuando como uma fonte de alimentação muito pequena, fornecendo uma quase perfeita tensão DC ininterrupta[11]. Se a tensão cair de repente, o capacitor usa sua energia carregada para fornecer a tensão adequada. O que acontece é que uma lógica rápida de troca de sinais entre valores lógicos altos e baixos pode drenar uma quantidade alta de potência em um período extremamente curto, da ordem de nanossegundos. A fonte de alimentação do circuito não consegue responder de modo rápido o suficiente para regular isso, e um capacitor de desacoplamento acaba impedindo que esse ruído de alta frequência seja percebido[11]. Em [11], o autor Jack Ganssle mostra que uma boa prática de projeto é colocar um capacitor de *Bypass* o mais próximo possível de cada CI (Circuito Integrado), sempre entre a alimentação e o terra.

2.3 Processamento em Tempo Real

Um sistema operacional em tempo real, ou RTOS (do inglês, *Real Time Operating Systems*), é atualmente a chave para o funcionamento de muitos sistemas embarcados na atualidade e consiste em uma plataforma de *software* sobre a qual muitas aplicações são construídas. Segundo [10], sistemas RTOS são uma categoria especial de programas determinísticos que conseguem organizar a execução de tarefas através de um gerenciamento ou escalonamento, de modo a garantir que todos os requisitos de tempo sejam cumpridos e, assim, que todas as tarefas sejam corretamente executadas dentro de seus prazos. São usados em aplicações que exigem confiabilidade e que tenham tempo de execução crítico. No livro [11] são citadas duas categorias de sistemas em tempo real:

- ***Soft Real-time***: aplicações em que o tempo de execução é crítico, mas possíveis atrasos não atrapalham o funcionamento do sistema, apenas degradam a qualidade da resposta;
- ***Hard Real-time***: aplicações em que o tempo de execução é extremamente crítico e caso os prazos não sejam cumpridos, o funcionamento do sistema pode ser comprometido.

Diante dessas duas definições, podemos classificar uma aplicação de robótica aérea como um sistema em tempo real do tipo *hard*. Isso porque uma falha na execução das tarefas no seu tempo certo pode causar um acidente, podendo levar à queda e consequente destruição da aeronave.

No contexto de aplicações em tempo real, surge a necessidade de usar funções que consigam ser executadas de forma concorrente compartilhando espaço de memória, podendo assim acessar variáveis uma da outra. Para esses tipos de procedimentos se dá o nome de *Threads*, que existem na forma de interrupções e conseguem ser executadas ao mesmo tempo. A alma do seu funcionamento é um escalonamento para decidir qual deve ser executada e quando o processador irá executá-la. Existem diversas formas de fazer essa distribuição estabelecendo prioridades, vendo seus *deadlines* e analisando o seu período de execução, para o caso de tarefas cíclicas. Quando todas as *Threads* possuem prioridades iguais de execução, dá-se o nome de princípio da equidade[12].

2.4 Comunicação SPI

Do inglês *Serial Peripheral Interface*, SPI é um protocolo de comunicação síncrono serial do tipo *full duplex* criado pela *Motorola* para promover uma simples interface entre diferentes microcontroladores e dispositivos [3]. O termo *full duplex* se refere ao fato de possuir linhas separadas para receber e enviar dados, o que significa que eles podem fluir nas duas direções simultaneamente. Todas as transmissões são sincronizadas por um mesmo relógio, ou *clock* em inglês, definido pelo mestre, e por isso ele é síncrono, diferente de uma porta serial padrão que é assíncrona. Algumas de suas vantagens são sua flexibilidade, sua simplicidade, sua capacidade de enviar e receber ao mesmo tempo, sua velocidade e sua grande utilização em diferentes aplicações.

O protocolo SPI incorpora uma arquitetura Mestre-Escravo, na qual o Mestre inicia, estabelece os parâmetros e controla toda a comunicação, que é baseada em 4 sinais principais:

- **MOSI**: do inglês *Master Out Slave In*, usada quando um dado é enviado do Mestre para o Escravo;
- **MISO**: do inglês *Master In Slave Out*, utilizada se o dado é enviado do Escravo para o Mestre;
- **SCLK ou SCK**: do inglês *Serial Clock*, sinal responsável por definir o relógio na comunicação, definido apenas pelo Mestre;
- **CS ou SS**: do inglês *Chip Select* ou *Slave Select*, sinal que escolhe o Escravo que deve receber ou enviar informações, já que mais de um dispositivo pode estar conectado à mesma interface SPI.

O Mestre controla a transmissão de dados pelo relógio e o Escravo usa esse sinal para sincronizar a sequência de *bits* que está sendo recebida. Toda vez que o Mestre pulsa o sinal na linha SCK, um *bit* de informação é empurrado para o Escravo e outro retorna para o Mestre. Os dados podem ser organizados em pacotes de 8 *bits*, ou um *byte*. Dessa forma, o *byte* é armazenado nos registradores de dados, e a pulsação do sinal de *clock* oito vezes pelo Mestre envia o pacote para o Escravo. A ideia para uma boa comunicação entre os dispositivos é saber de antemão quando o Escravo retornará os dados, e o quanto será retornado. A Figura 2.2 mostra um exemplo básico de interface SPI.

2.5 Comunicação I^2C

Do inglês *Inter-Integrated Circuit*, o protocolo de comunicação I^2C é uma interface serial baseada em dois fios desenvolvido pela *Philips*, que permite a comunicação entre diferentes circuitos integrados, ou dispositivos. Utilizando uma interface que permite múltiplos Mestres e Escravos com detecção de colisão, em uma mesma rede pode conter até 128 dispositivos [13], cada um com seu próprio endereço fixo podendo ser configurado para transmitir ou receber dados. A

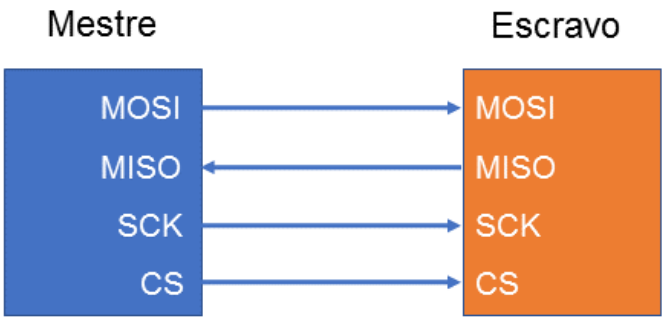


Figura 2.2: Interface básica SPI (Adaptado de [3]).

transmissão da informação usa duas linhas bidirecionais que conectam os componentes em um mesmo barramento, são elas:

- **SCL**: do inglês *Serial Clock Line*, consiste em uma linha que varia o sinal do *clock* entre baixo e alto;
- **SDA**: do inglês *Serial Data Line*, é a linha responsável por transmitir os dados entre os Mestres e os Escravos.

A comunicação I^2C é bem mais lenta que a SPI, mas sua aplicação é muito difundida no mercado, sendo utilizado em uma grande quantidade de componentes eletrônicos principalmente pelo seu custo e fácil implementação. A Figura 2.3 mostra um exemplo básico de interface I^2C.

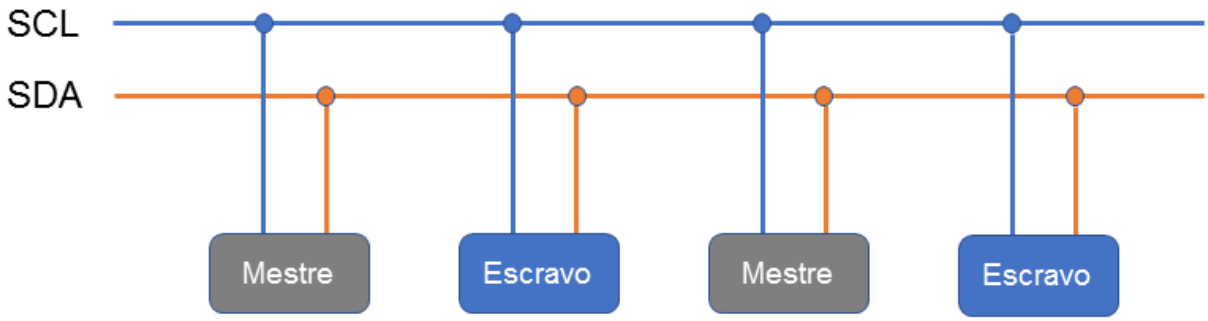


Figura 2.3: Interface básica I^2C (Adaptado de [3]).

2.6 Sensores

Um sensor é um dispositivo que detecta ou mede uma quantidade física; este trabalho atentou-se apenas com os de saída elétrica.

2.6.1 Unidade de Medição Inercial

Uma Unidade de Medição Inercial, ou IMU (do inglês, *Inertial Measurement Unit*), é um conjunto de sensores que inclui: um acelerômetro de 3 eixos e um girômetro de 3 eixos. O modelo da *Adafruit* utilizado neste projeto possui ainda um magnetômetro e um barômetro, formando no total 10 graus de liberdade, por isso seu nome 10 DOF, sigla para *Degrees of Freedom*. Uma IMU contem boa parte da instrumentação necessária para um veículo aéreo não tripulado. Embora toda a implementação de *hardware* e *software* para a utilização de todos os sensores tenha sido feita, o foco deste trabalho será o controle de atitude do veículo aéreo baseado no funcionamento do girômetro.

O girômetro é basicamente um giroscópio de medição implementado em um sistema microeletromecânico, mais conhecido como MEMS (do inglês, *Microelectromechanical Systems*). Este sistema é formado por transdutores com estruturas micromecânicas integradas a elementos elétricos. O modelo utilizado neste projeto é o L3GD20H que possui 3 eixos e realiza a medição da velocidade angular em cada um deles. A rotação do corpo cria uma força de Coriolis que oscila um par de estruturas mecânicas microscópicas com uma mesma amplitude e direções opostas. A amplitude da oscilação é diretamente proporcional à taxa de variação angular. Quando esses corpos sofrem uma rotação, a força de Coriolis cria uma vibração ortogonal ao eixo de oscilação. Essa vibração fora do plano de oscilação pode ser medida através da variação capacitiva dos elementos da estrutura[5].

2.6.2 Sensor Ultrassônico

De acordo com [14], sensores ultrassônicos, ou sonares, usam ondas acústicas para a detecção de objetos em distâncias que podem chegar em mais de 12 metros. Um trem de pulsos excita um transdutor que gera uma onda de pressão que se propaga no meio até atingir o alvo. Parte da sua energia retorna em forma de um eco após um intervalo de tempo. Estes sensores são encontrados em inúmeras aplicações em indústrias e em sistemas de segurança; em aplicações de robótica aérea são usados normalmente para medir a altitude da aeronave em relação ao solo. Existem diversos tipos no mercado e o modelo utilizado neste trabalho tem seu funcionamento baseado na emissão de uma onda sonora em uma frequência de 40kHz, que é refletida de volta na mesma direção ao se deparar com algum obstáculo. Sabendo que a velocidade de propagação do som no ar é de 340 m/s, para calcular a distância basta medir o tempo que a onda demorou para ir e retornar e dividir por 2, pois ela percorre duas vezes o mesmo percurso a ser medido. A expressão matemática é dada então por

$$h = \frac{tv_s}{2}, \tag{2.1}$$

sendo h a distância do anteparo, t o tempo em segundos que a onda mecânica gastou para ir e retornar e v_s a velocidade de propagação do som no ar. A Figura 2.4 ilustra o funcionamento descrito do sensor.

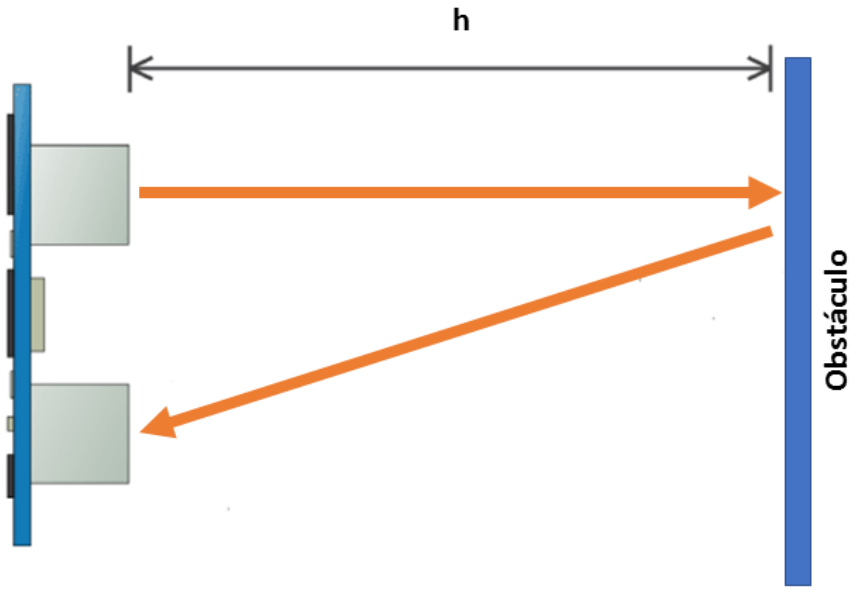


Figura 2.4: Funcionamento do sensor ultrassônico modelo HC-SR04.

O modelo HC-SR04 utilizado possui 4 pinos, são eles:

- **VCC**: pino de alimentação;
- ***Trigger***: sinal de ativação. Emite um pulso de 10μs que faz com que o módulo envie uma sequência de 8 ciclos de onda ultrassônica a uma frequência de 40kHz cada;
- ***Echo***: sinal de retorno, que permanece em nivel alto até a onda sonora retornar;
- **GND**: pino de referência, ou terra do componente.

A Figura 2.5 mostra o diagrama de tempo encontrado no manual do sensor[1], que ilustra o funcionamento dos pinos *Trigger* e *Echo*.

Existem dois parâmetros que definem um sensor ultrassônico, a propagação da onda sonora no meio em que foi emitida e o seu ângulo de efeito[14]. Fatores como a densidade do ar e a sua temperatura podem influenciar na velocidade da onda e alterar a precisão do sensor. Já o ângulo de efeito corresponde à inclinação mínima em que o sensor deve estar posicionado de modo que a onda refletida consiga retornar.

2.7 Atuadores

Ao contrário dos sensores, os atuadores convertem um sinal, geralmente elétrico, em alguma ação mecânica. São dispositivos capazes de transformar a energia de entrada em alguma forma de atuação no sistema.

[1]Manual do sensor ultrassônico HC-SR04 disponível em: <http://www.micropik.com/PDF/HCSR04.pdf>. Acesso em: 24 de novembro de 2017.

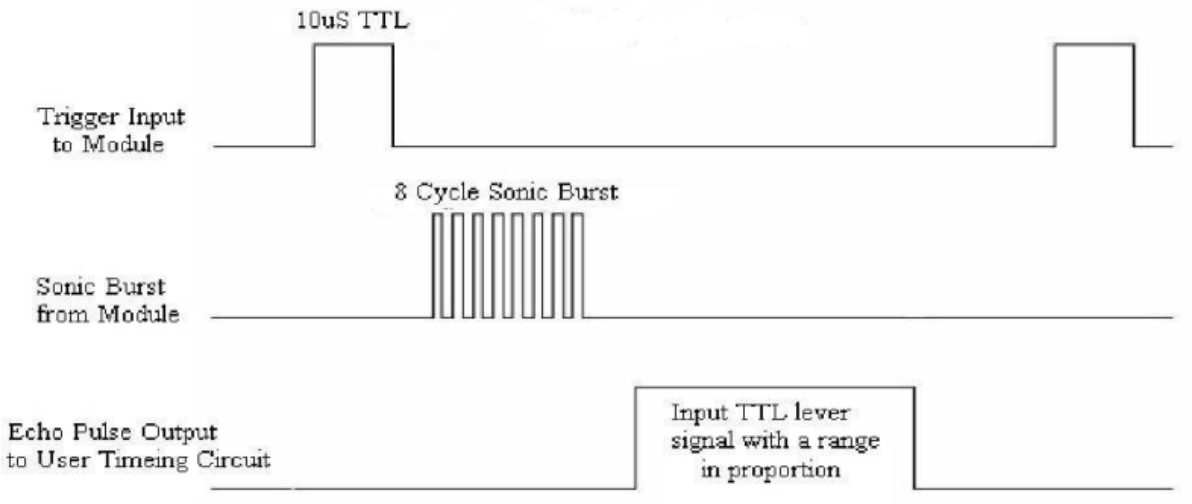


Figura 2.5: Diagrama de tempo do sensor ultrassônico modelo HC-SR04 (Fonte: retirado do manual do sensor).

2.7.1 Servomotor

Um servomotor é um tipo de motor DC especialmente desenvolvido com um sistema de retroalimentação. São dispositivos eletromecânicos usados em diversas aplicações que exigem movimentos precisos e controlados. Como mostrado na Figura 2.6, seu interior é composto por: um potenciômetro, ligado ao eixo que monitora a sua posição; um circuito de controle, responsável pelo posicionamento angular do eixo feito através do monitoramento do potenciômetro e acionamento do motor; um motor DC; e um conjunto de engrenagens, que atuam como um redutor, transferindo mais torque ao eixo. O comando de posição é enviado ao servo por um sinal PWM (do inglês, *Pulse-Width-Modulation*).

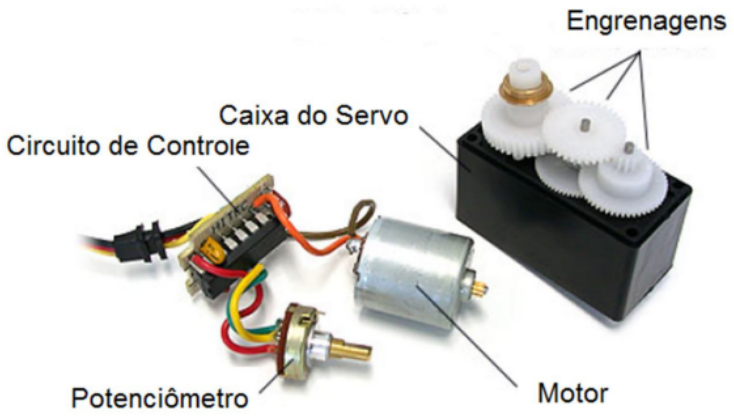


Figura 2.6: Componentes internos de um servomotor.

PWM, ou Modulação por Largura de Pulso, é uma técnica muito usada para codificar um sinal analógico em um sinal digital, permitindo controlar diversos dispositivos. Seu princípio de funcionamento consiste na modificação, ou modulação, de uma onda retangular periódica digital

fazendo-a ter diferentes larguras de pulso (*duty cycle*, em inglês)[4], e esta irá conter a informação. A Figura 2.7 mostra um exemplo de sinal PWM. Comparando as duas formas em (a) e (b), os dois sinais possuem o mesmo período e frequência, porém o valor médio da saída M_1 é menor que M_2, visto que S_1 e S_2 possuem diferentes larguras de pulso, 25% e 50%, respectivamente. Portanto, um circuito com PWM pode ser facilmente programado para modificar a sua largura do pulso alto, quanto maior for, maior será a tensão de saída. No caso de um servomotor, a posição angular de seu eixo estará diretamente relacionada com a largura do pulso enviado, monitorada em ciclos de 20ms. Se nesse intervalo de tempo, alguma mudança for detectada, sua posição é alterada.

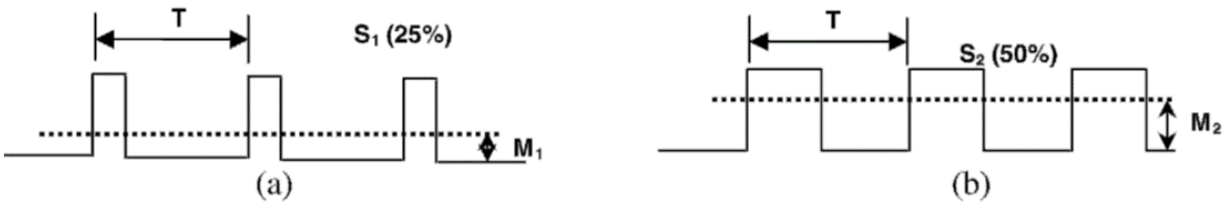


Figura 2.7: Um exemplo de sinal PWM (Fonte: retirado de [4]).

2.7.2 Motor C.C. sem Escovas

Motores C.C. sem escovas (do inglês, *Brushless Motors*) são motores síncronos de corrente contínua que funcionam através de um inversor ou de uma comutação na alimentação. Dessa forma, uma corrente elétrica alternada é gerada para atuar em cada fase do motor por meio de um controlador em malha fechada. Este fornece pulsos de corrente para os enrolamentos do motor controlando a sua velocidade e seu torque. O módulo que realiza o controle e acionamento de um motor sem escovas no contexto de aeromodelismo é chamado de ESC (do inglês, *Electronic Speed Controller*).

Em um típico motor C.C., existem ímãs permanentes na parte externa que são estacionários, por isso chamados de estatores, e uma armadura que gira na parte interna, que recebe o nome de rotor. Uma maior confiabilidade, um ruído reduzido, uma vida útil mais longa (por não possuírem escovas, que desgastam muito), uma redução de interferência eletromagnética, um menor peso e, principalmente, uma maior eficiência na conversão de energia elétrica em mecânica, são exemplos de vantagens na utilização de motores sem escovas. Sua desvantagem seria o elevado custo de implementação em um projeto, o que atualmente já não é um grande problema, pois há no mercado, cada vez mais modelos acessíveis com preços menores.

2.8 Sistemas de Controle

De acordo com as definições apresentadas por Ogata[15], um sistema de controle com realimentação consiste em um sistema que mantém uma relação entre a sua saída e a sua entrada de

referência, por meio da comparação e da diferença entre essas duas informações. Um exemplo que o autor cita é o controle de temperatura em um quarto, sendo que a referência seria a temperatura desejada e a saída seria a temperatura atual medida neste ambiente. Dependendo da diferença entre esses valores, o sistema controlará o termostato para aquecer ou resfriar o quarto, até garantir que a temperatura no quarto chegue ao valor desejado, independente de condições externas. A diferença entre o sinal de entrada e o sinal de realimentação é chamado de erro do sistema. Em um sistema de controle em malha fechada, esse erro é alimentado ao controlador até que seja reduzido, levando o valor de saída a atingir o valor desejado.

Quando se fala de controle em malha fechada, sempre haverá uma ação de realimentação para reduzir o erro do sistema. Diferente do caso de um sistema de controle em malha aberta, no qual a saída não tem nenhum efeito na ação de controle, ou seja, não existe uma realimentação e nem uma comparação para verificar a saída em relação ao valor de referência desejado. Dessa forma, cada entrada corresponde a uma condição de operação fixa, o que faz com que a precisão do sistema dependa da sua calibração. A presença de distúrbios externos afeta diretamente a resposta de um controle em malha aberta, fazendo com que seja recomendado apenas se a relação entre a entrada e a saída do sistema for bem conhecida. Já em um controle de malha fechada, o fato de possuir uma realimentação o torna mais robusto e menos sensível a fatores externos e/ou internos. Entretanto também torna mais difícil para o sistema de se obter estabilidade, já que sua tendência de corrigir os erros pode causar mudanças de amplitude ou oscilações na resposta. Portanto, para sistemas nos quais as relações entre as entradas e as saídas são bem conhecidas e não há distúrbios, é recomendado um controle em malha aberta, enquanto isso, um controle em malha fechada será a melhor opção quando existirem variações e distúrbios imprevisíveis. A Figura 2.8 retrata o diagrama de blocos dos dois tipos de controle explicados. Como explicado, é possível ver que, no controle em malha fechada, o erro é expresso na forma

$$e(t) = r(t) - y(t), \tag{2.2}$$

sendo $e(t)$ o erro de controle no instante de tempo t, calculado pela diferença entre o valor de referência ou desejado (sinal de entrada), $r(t)$, e o valor medido pelo sistema (sinal de realimentação), $y(t)$.

Em um sistema, podemos implementar ações de controle Proporcional(P), Derivativa(D) e Integral(I). Cada uma dessas ações possui um efeito e pode ser usada individualmente ou combinada às outras de diferentes modos. Aplicando as três juntas obtemos um controlador Proporcional-Integral-Derivativo, mais conhecido como PID. O controlador PID e suas variações são os algoritmos mais conhecidos e utilizados em sistemas de controle com realimentação atualmente[16]. Neste trabalho foram utilizados apenas controladores do tipo Proporcional (P), e Proporcional-Derivativo (PD), mas estão aqui brevemente explicadas cada uma dessas ações de controle.

- **Ação Proporcional**

A ação de controle Proporcional possui uma característica na qual o controlador é proporcional ao erro medido do sistema. Dessa forma, podemos expressá-lo como

$$u(t) = K_P e(t). \tag{2.3}$$

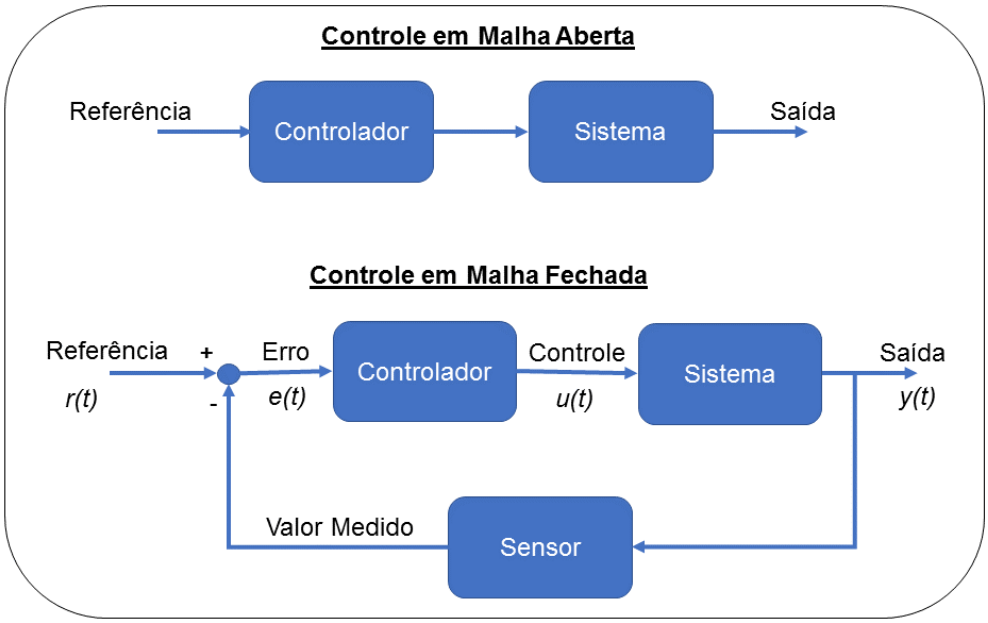


Figura 2.8: Controle em Malha Aberta e em Malha Fechada.

A variável $u(t)$ se refere ao sinal de controle, obtido multiplicando o ganho proporcional K_P pelo erro. Dessa forma, quanto maior a distância entre o valor medido e o de referência, maior será a ação de controle para tentar compensar isso[17]. Quanto maior o ganho, menor o erro em regime permanente, porém se aumentarmos muito o ganho pode aumentar também a resposta oscilatória do sistema levando-o à instabilidade. Caso o ganho seja baixo, a resposta pode não atingir o valor desejado, aumentando o erro em regime permanente.

- **Ação Derivativa**

A ação de controle Derivativa é baseada na predição de valores futuros do erro, por isso é chamada também de controle antecipado[17]. Podemos expressá-la como

$$u(t) = K_D \frac{de(t)}{dt}, \tag{2.4}$$

sendo $u(t)$ o sinal de controle, obtido multiplicando o ganho derivativo K_D pela derivada do erro. A ação derivativa nunca é implementada sozinha, mas quando implementada com um controlador proporcional fornece um meio de se obter um controlador com alta sensibilidade. Sua vantagem é que faz com que a ação de controle responda de acordo com a taxa de variação do erro, produzindo uma correção antes que a magnitude do erro se torne muito grande[15], tendendo a aumentar a estabilidade do sistema. Embora o controle derivativo não afete diretamente o erro em regime permanente, ele adiciona uma oscilação ao sistema, permitindo o uso de valores altos de K_D melhorando o erro em regime permanente.

- **Ação Integral**

A ação de controle Integral é proporcional à integral do erro, podendo ser expressada como

$$u(t) = K_I \int_0^t e(\tau)d\tau, \tag{2.5}$$

sendo $u(t)$ o sinal de controle, obtido multiplicando o ganho integrativo K_I pela integral do erro. Enquanto a ação proporcional é baseada nos valores presentes e a derivativa é baseada em valores futuros, a integral é baseada em valores passados do erro. A principal vantagem da componente integral é que ela permite a redução do erro em regime permanente para zero.

Apresentada as três ações de controle, podemos definir a ação de controle de um controlador Proporcional-Derivativo da seguinte forma

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt}, \tag{2.6}$$

e de maneira semelhante definir a ação de controle para um controlador Proporcional-Integral-Derivativo como

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}. \tag{2.7}$$

Capítulo 3

Desenvolvimento

3.1 Introdução

O desenvolvimento de um sistema robótico aéreo de dois rotores articulados possui uma elevada complexidade. Isso, porque envolve a aplicação de conhecimentos de três grandes áreas da engenharia: a mecânica, a elétrica e a computação. O trabalho iniciado por Davi Feques Vale e Mickael Ângelo A. da Costa [5] deu origem ao primeiro veículo aéreo não tripulado com dois rotores, desenvolvido no Laboratório de Robótica Aérea da Universidade de Brasília. Um primeiro passo foi dado e neste capítulo será apresentado o segundo, detalhando todo o projeto de *hardware* e *software* dessa continuação, que visa tornar o sistema birrotor algo mais completo e próximo de um produto fechado.

3.2 *Hardware*

3.2.1 Definição da Topologia do Sistema

O projeto da eletrônica embarcada era composto por dois módulos: um módulo de potência (ainda não implementado, vide [5], página 87), e um módulo de aquisição e processamento de dados. O primeiro, formado por uma bateria e uma placa de distribuição de energia, e o segundo, formado pelo microcontrolador e sua interface com o computador pessoal e o *joystick*. Como processador central, estava sendo utilizado um *Arduino Pro Mini*, o qual era responsável por quase todas as tarefas, mais especificamente, por obter os dados dos diferentes componentes eletrônicos, enviar as informações obtidas via cabo USB para um computador pessoal com plataforma *Windows*, e também por enviar os sinais PWM para os dois servo motores e os dois ESCs. A topologia descrita do sistema antigo é apresentada na Figura 3.1.

Entretanto, essa configuração possui limitações, principalmente devido às conexões cabeadas com o computador pessoal, que impossibilitam a realização de testes adequados no veículo aéreo e, o mais importante, impedem que ele voe. Outra limitação é a capacidade de processamento do *Arduino*, pois existe a necessidade de um processador com maior capacidade para a implementação

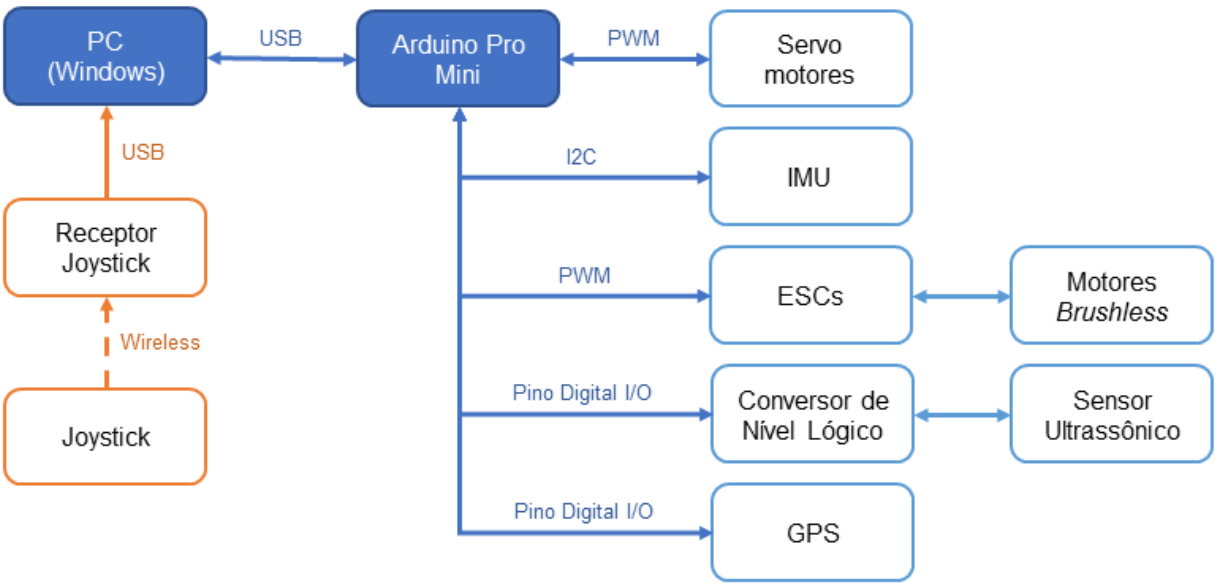


Figura 3.1: Topologia inicial do sistema birrotor (Fonte: adaptado de Davi e Mickael, 2016 [5]).

de algoritmos de controle e de estimação, ou outros algoritmos complexos presentes neste vasto alcance de aplicações de robótica aérea. Um ponto importante a se destacar é o elevado custo de comunicação entre os diferentes componentes do sistema; o *joystick* se comunica primeiro com o computador pessoal através de um programa chamado *Processing* baseado em linguagem Java, para então realizar a troca de informações com o *Arduino* via serial. Com tudo isso, a topologia precisava ser revista.

Tendo disponível no laboratório o microcontrolador *Gumstix*[1], este então faria o trabalho de processamento central, recebendo os dados e distribuindo os comandos, enquanto o *Arduino* estaria conectado com os outros componentes, com exceção do módulo GPS. Ambos os microcontroladores se comunicariam através de uma comunicação SPI e, assim, o *Arduino* seria apenas um intermediador recebendo ordens do microcontrolador principal.

Um longo período de tempo foi gasto estudando as placas *Gumstix* para compreender melhor seu funcionamento. A primeira pergunta feita foi qual dos modelos utilizar para realizar uma função tão importante para o sistema. No laboratório havia disponível dois modelos, uma *Overo Fire*[2] [18] e uma *Overo Water*[3][19]. Optou-se pela utilização do modelo *Overo Fire* por possuir um módulo de comunicação sem fio, isso juntamente com uma placa de expansão *Tobi*[4] acoplada, que possui periféricos USB necessários para conectar o *joystick* e uma quantidade grande de pinos de entrada e saída para interface com componentes eletrônicos, mais especificamente no nosso caso o

[1] *GUMSTIX, INC. Gumstix, dream, design, deliver.* Disponível em: <https://www.gumstix.com/>. Acesso em: 03 de novembro de 2017.

[2] *OVERO FIRE COM.* Datasheet disponível em: <https://s3-us-west-2.amazonaws.com/media.gumstix.com/datasheets/GUM3503F.pdf>. Acesso em: 03 de novembro de 2017.

[3] *OVERO WATER COM.* Datasheet disponível em: <https://s3-us-west-2.amazonaws.com/media.gumstix.com/datasheets/GUM3503W.pdf>. Acesso em: 03 de novembro de 2017.

[4] *TOBI. Expansion Board for Gumstix.* Disponível em: <https://store.gumstix.com/tobi.html>. Acesso em: 03 de novembro de 2017.

GPS. Seguindo as intruções da própria página do fabricante, o primeiro procedimento foi instalar uma imagem Linux em um cartão micro SD de no mínimo 2GB. Existem dois métodos para isso: o primeiro, e bem mais demorado, é construir a imagem do zero; enquanto o segundo, é baixar uma imagem pronta e copiá-la para o cartão de memória. Enormes dificuldades foram encontradas logo nesse primeiro passo, por mais de um mês várias tentativas foram realizadas, porém sem sucesso. O tempo de construção de uma imagem é relativamente alto, o que atrapalhava muito. Após algum tempo insistindo, descobriu-se que trocando a placa pela *Overo Water*, as imagens inseridas funcionavam perfeitamente. Não se sabe ao certo, mas acredita-se que por ser uma versão mais antiga (não é mais comercializada nem na própria loja do fornecedor), a *Overo Fire* não tenha compatibilidade com imagens de sistemas operacionais mais recentes. Uma imagem do sistema operacional Ubuntu, em sua versão 16.04, foi criada no cartão SD e funcionou perfeitamente na placa.

Durante os primeiros meses deste trabalho, a placa *Gumstix* foi utilizada. Contudo, apesar de neste pequeno período ter proporcionado um bom aprendizado em sistemas embarcados, a dificuldade de encontrar uma boa documentação e o consequente atraso no desenvolvimento da plataforma do veículo áereo acabou levando a uma decisão importante no projeto. Houve uma troca por um modelo de outro fabricante, o *Raspberry Pi 3* Modelo B[5]. Obteve-se, então, uma nova topologia para o sistema, apresentada na Figura 3.2, a qual seria utilizada durante o resto do desenvolvimento do projeto.

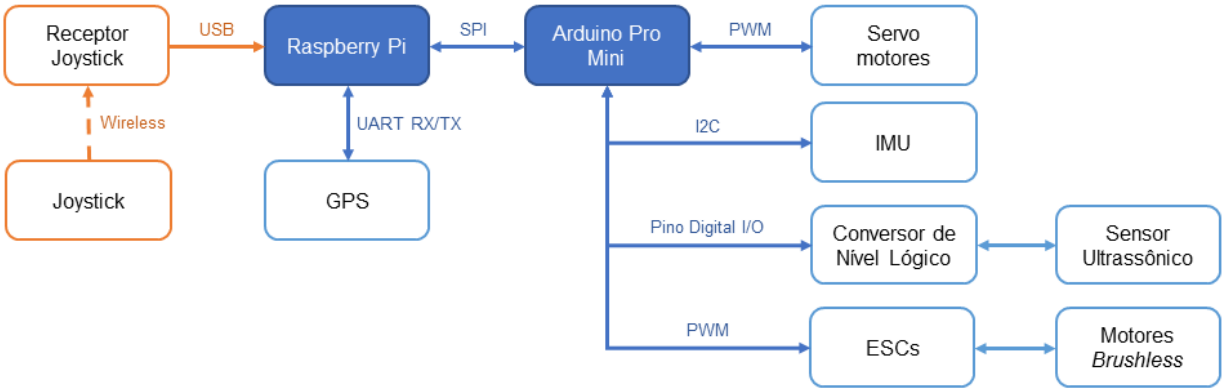


Figura 3.2: Topologia nova do sistema birrotor.

3.2.2 Projeto de um Módulo Medidor de Tensão

Um sistema robótico aéreo não tripulado precisa ter a bateria de alimentação do sistema monitorada constantemente, visto que a situação na qual a carga da bateria termina em pleno voo pode levar à queda do veículo com consequências graves para a integridade do conjunto como um todo. Pensando nisso, foi realizado o projeto de um módulo para a eletrônica embarcada que realizasse a medição da tensão fornecida pela bateria. O circuito seria basicamente um divisor

[5] *Raspberry Pi 3 Model B*. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: 03 de novembro de 2017

de tensão, cuja saída seria controlada por uma porta analógica do *Arduino*, esta faria então a conversão analógica/digital (A/D) do valor lido. A Figura 3.3 mostra o esquemático do circuito feito no *software* de simulação *Proteus*[6], considere os nomes utilizados nele para identificar seus elementos.

Alguns aspectos de projeto precisavam ser levados em consideração:

1. A tensão especificada na bateria Lipo a ser utilizada é de 11,1V, mas como medida de proteção, para um caso extremo foi assumido um valor máximo de 15,0V;
2. A tensão máxima permitida em uma porta analógica do *Arduino Pro Mini* corresponde ao seu VCC, no caso 3,3V. Para garantir uma faixa de segurança, assumiu-se que a tensão na porta analógica, chamada aqui de V_{ADC}, variaria entre 0V e 1,1V, aproximadamente;
3. A corrente precisa ser pequena para não haver um consumo grande de energia. Assumiu-se, então, uma corrente total chamada de I, que deve ser menor que 1mA;
4. O resistor R_1 portanto deve ter uma resistência alta;
5. Um capacitor deve ser colocado na entrada analógica para evitar picos e erros na leitura do conversor A/D do microcontrolador, de modo que a constante de tempo do circuito fique menor que 100ms.

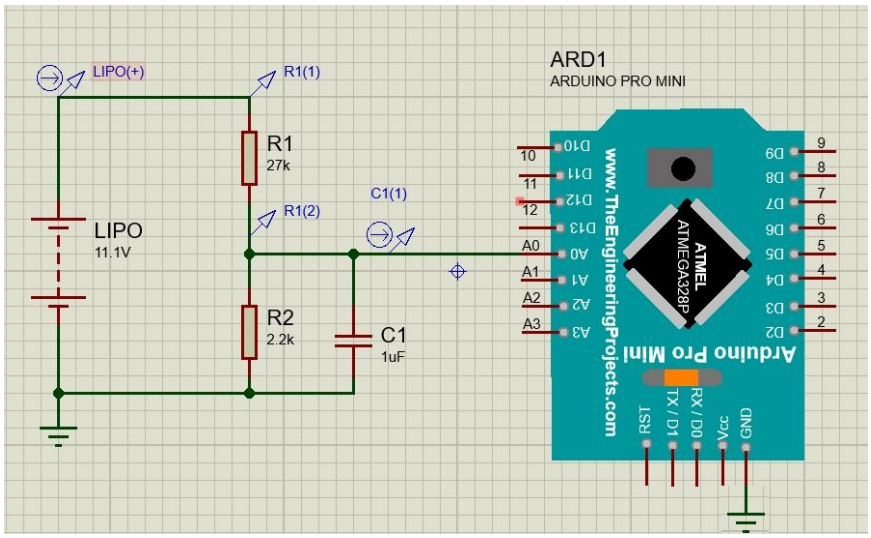


Figura 3.3: Circuito medidor de tensão da bateria.

Diante desses requisitos estabelecidos, e observando os valores comerciais de resistores e capacitores, foi definido para R_1 um valor de $27k\Omega$. Para determinar R_2, sabemos pela equação do divisor de tensão que

$$V_{ADC} = \frac{R_2}{R_1 + R_2} V_{bateria}. \tag{3.1}$$

Com isso, podemos analisar para o limite superior definido, no qual temos como máximo, uma tensão de 15,0V na bateria e na porta analógica, uma tensão V_{ADC} igual a 1,1V. Calculando,

[6] *Proteus PCB Design and Simulation.* Disponível em: <https://www.labcenter.com/>.

obtemos que R_2 possui um valor aproximado de $2,2k\Omega$. A Figura 3.4 mostra o diagrama de simulação do circuito, indicando uma tensão de saída V_{ADC} de 1,13V para uma tensão de entrada limite de 15,0V, com uma corrente total de aproximadamente 0,5mA. Confirmando assim, os requisitos de projeto estabelecidos.

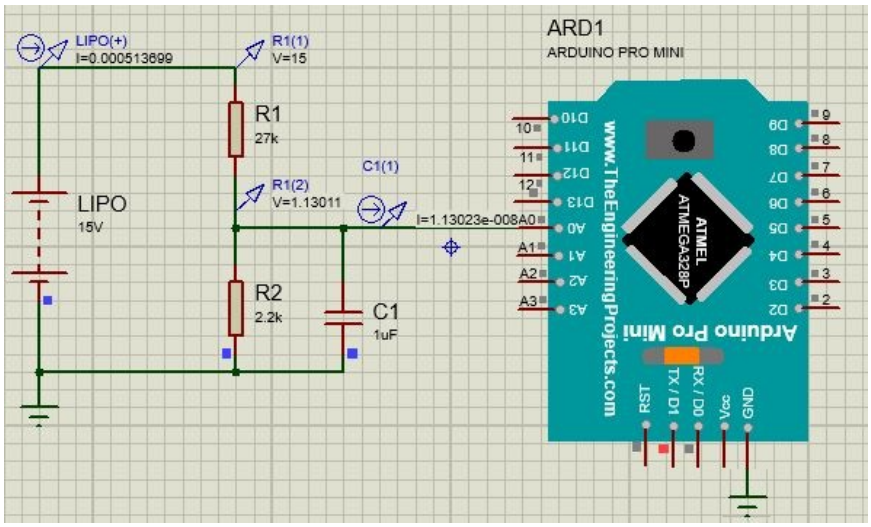


Figura 3.4: Simulação do circuito medidor de tensão.

Para fornecer uma tensão DC ininterrupta e impedir que o ruído de alta frequência seja percebido, foi definido como requisito de projeto que a constante de tempo τ fique menor que 100ms, de modo que o produto $(R_1||R_2)\times$C seja menor que 100ms, sendo $(R_1||R_2)$ a associação em paralelo dos dois resistores. Portanto, o valor definido para o capacitor foi de $1\mu F$. Os principais valores correlacionados obtidos na simulação são mostrados resumidos na Tabela 3.1.

Tabela 3.1: Valores de Tensão Obtidos na Simulação

$V_{Bateria}$	V_{ADC}
15,0V	1,13V
11,1V	0,83V
0V	0V

3.2.3 Projeto Eletrônico do Sistema

O circuito inicial do sistema apresentava problemas técnicos, entre eles cabe destacar: a presença de fios finos que facilmente se rompem com a movimentação da placa; a falta de um esquemático que representasse todos os módulos conectados na placa de aquisição de dados; e a falta de um projeto da placa com as trilhas representadas. Ademais, toda a solda foi feita em uma placa do tipo universal[7] e não possuía nenhum arquivo de projeto feito em algum tipo de programa de

[7]Placa Universal: placa utilizada para prototipagem rápida de circuito que já vem toda perfurada.

modelagem de circuitos ou de modelagem de placas de circuito impresso[5].

A Figura 3.5 mostra uma foto do circuito encontrado, enquanto a Figura 3.6 mostra sua vista inferior. Nas duas fotos é possível visualizar finos fios vermelhos passando de um lado para o outro, que além de provocarem ruídos de indutância no circuito também são facilmente rompidos, especialmente quando se trata de um veículo aéreo. Destaca-se também a ausência da placa de potência, como explicado pelo próprio trabalho desenvolvido em [5]. Ela seria responsável pela distribuição de energia da bateria entre os diversos componentes que possuem diferentes tensões de alimentação.

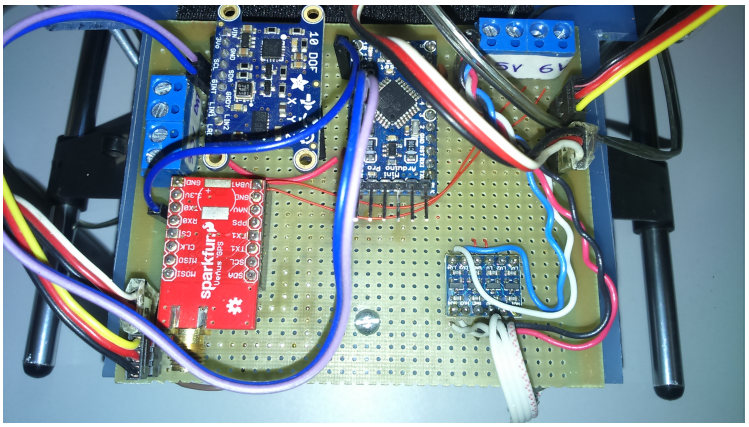


Figura 3.5: Circuito antigo do sistema birrotor.

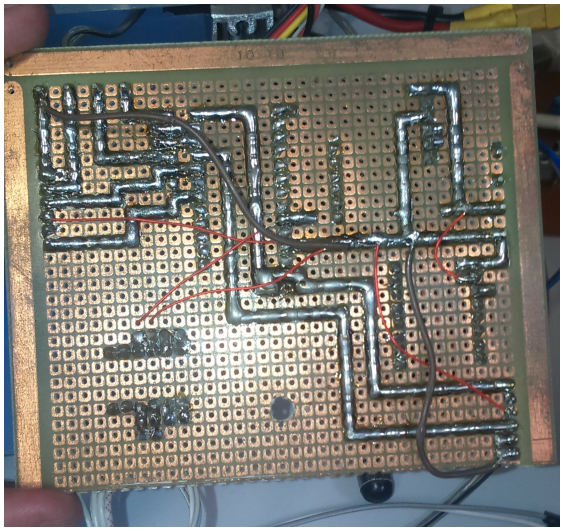


Figura 3.6: Trilhas soldadas do circuito da Figura 3.5.

Primeiramente, todos os módulos sensores, atuadores e processadores foram estudados de modo individual para se compreender a estrutura e a necessidade física de cada um. Questões referentes às tensões de alimentação, corrente e potência máximas permitidas, e exemplos de aplicações, foram analisadas caso a caso através de manuais ou *datasheets* fornecidos pelos fabricantes. Com base no trabalho desenvolvido anteriormente e a partir de estudos realizados, foi feito um processo de engenharia reversa para identificar todas as trilhas e conexões do circuito.

Considerando tudo o que foi exposto, foram definidos pontos importantes para o novo projeto eletrônico do sistema. Existem no veículo áereo duas frentes que tinham a necessidade de se manter separadas, uma que envolve alta corrente e outra que envolve uma corrente inferior. Assim, definiu-se que o projeto seria dividido em dois módulos, um operando em alta corrente e o outro em uma corrente mais baixa, divisão que pode ser vista na Figura 3.7. No primeiro, estariam conectados os motores sem escovas responsáveis pelo empuxo para retirar o veículo do chão e também os módulos de controle de velocidade desses motores (ESCs). Já no segundo, estariam os módulos sensores, os servo motores e os microcontroladores. Os servos e o microcontrolador principal consomem uma corrente considerável, entretanto estão nesse grupo por possuírem correntes bem inferiores, se comparados com os que se encontram no módulo de alta corrente. Dito isso, chamaremos esse grupo de módulo de baixa corrente. Dessa forma, temos três faixas de valores de tensão: 11,0V, 5,0V e 3,3V. A maior tensão alimenta diretamente os motores *Brushless*[8] e os módulos ESCs; a de 5,0V alimenta o *Arduino Pro Mini*, o *Raspberry Pi*, o sensor ultrassônico e ambos os servo motores; e por último, temos a tensão de 3,3V, que alimenta o módulo inercial (IMU) e o módulo GPS.

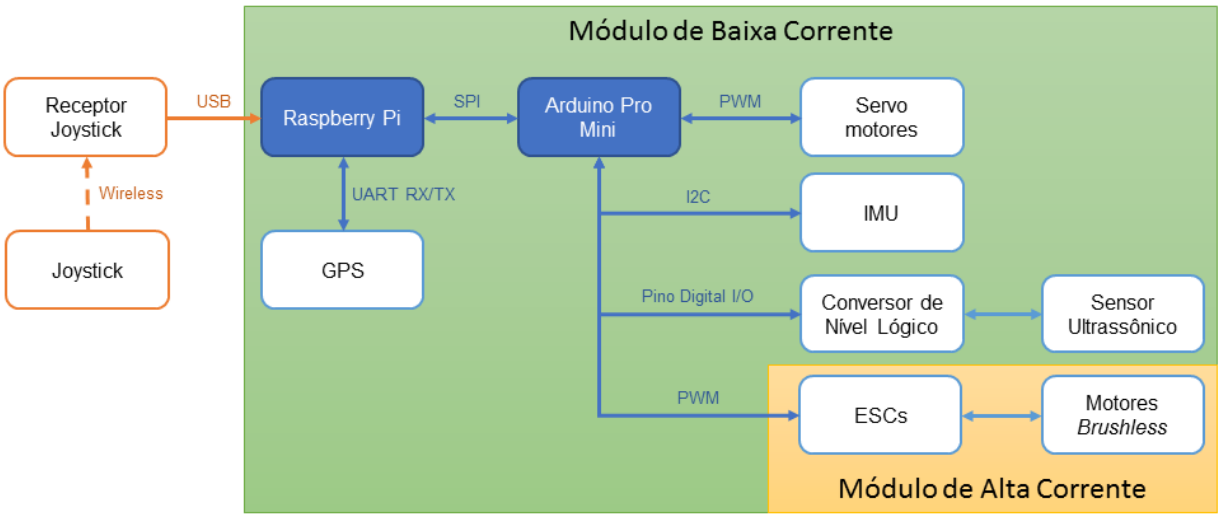


Figura 3.7: Divisão entre módulos de acordo com a corrente.

Substituindo a placa de potência sugerida em [5], foi inserido um conversor DC/DC do tipo *step-down* modelo LM2596[20] que faz a conversão da tensão de entrada de 11,1V da bateria Lipo e a regula para um valor ajustável de acordo com um *trimpot*, que é uma espécie de potenciômetro regulável em miniatura com um pequeno parafuso. Este foi então ajustado para fornecer 5,0V. Durante boa parte do desenvolvimento esse conversor foi utilizado, entretanto descobriu-se um problema. Apesar de na sua documentação indicar uma corrente de saída de no máximo 3A, observou-se na prática que com uma tensão de saída de 5,0V, a corrente não conseguia chegar a esse valor, e o que ela atingia não era suficiente para alimentar o circuito sem prejudicar a qualidade de seu funcionamento. Dependendo da situação, os servos não recebiam corrente suficiente para realizar o esforço necessário e começavam a tremer, ou paravam de responder. Por esse motivo,

[8] *Brushless*: sem escovas, traduzido do inglês.

um novo conversor foi adquirido, só que agora no modelo XL4015[21], que fornece uma corrente máxima de até 5A e também uma potência maior, com um máximo de 75W. Já para fornecer o outro nível de tensão correspondente a 3,3V, foi inserido um regulador de tensão simples do modelo LDV1117V33[22], que foi instalado com dois capacitores de desacoplamento, um de 100nF e outro de 10μF.

Diante do que foi descrito e com uma maior compreensão do sistema, foi possível obter o esquemático do circuito completo com um maior nível de detalhamento. A Figura 3.8 mostra o módulo de alta corrente, enquanto a Figura 3.9 mostra o módulo de baixa corrente (versão ampliada no Anexo I), ambos desenhados por meio de uma plataforma *web* chamada *Easy EDA*[9] indicada pela equipe do Laboratório de Robótica e Automação da Universidade de Brasília, o LARA[10]. Essa plataforma foi escolhida por permitir a criação de esquemáticos extremamente detalhados e por possuir uma biblioteca com uma enorme variedade de módulos e componentes eletrônicos; todos os utilizados para a criação do circuito proposto foram encontrados. Outro ponto positivo é que além de possuir uma interface extremamente amigável ao usuário, permite também fazer a criação e o projeto de uma placa de circuito a partir do esquemático criado, com todos os CADs com as dimensões apropriadas de cada componente. O circuito projetado foi discutido com o professor orientador do projeto Geovany Araújo Borges e modificado até chegar nessa versão aqui apresentada. Um outro esquemático mais simples visualizado na Figura 3.10 representa um circuito auxiliar feito para distribuir a energia entre os dois módulos e para ligar e desligar o sistema através de uma chave, com um LED azul de alto brilho indicando se está ligado ou não.

O *Arduino Pro Mini* aparece presente nos dois módulos, no de alta e no de baixa corrente. Em ambos é possível ver a representação de seus pinos de entrada e saída, tanto analógicos como digitais. De modo resumido, assim ficaram distribuídas suas conexões:

- Analógicas
  - **A0**: medição da tensão na bateria;
  - **A1**: acionamento do LED vermelho de alarme de bateria fraca;
  - **A4** e **A5**: SDA e SCL, respectivamente, responsáveis pela comunicação I2C com a unidade de medição inercial - IMU;
- Digitais
  - **D3**: acionamento via PWM do ESC esquerdo;
  - **D9**: acionamento via PWM do ESC direito;
  - **D5**: acionamento via PWM do servo motor esquerdo;
  - **D6**: acionamento via PWM do servo motor direito;

[9] *EASY EDA*. Plataforma online de desenvolvimento e simulação de circuitos. Dísponível em: <https://easyeda.com/>. Acesso em: 05 de novembro de 2017.

[10] LARA. Site do Laboratório de Automação e Robótica da UnB disponível em: <https://lara.unb.br/>. Acesso em: 07 de julho de 2017.

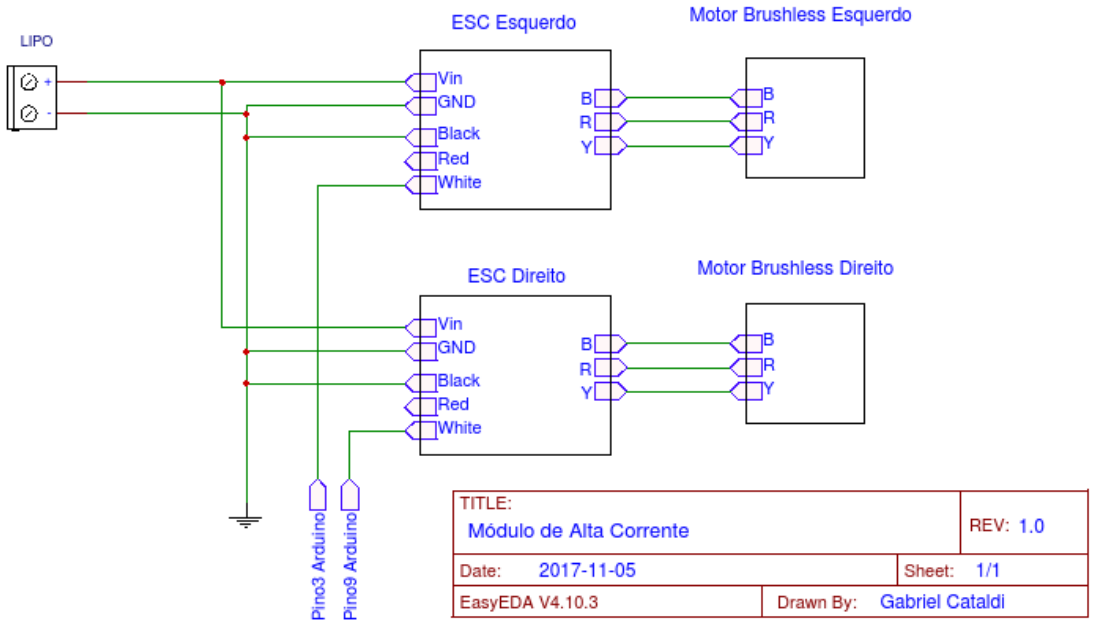


Figura 3.8: Esquemático detalhado do módulo de alta corrente.

- **D7**: sinal *thrigger* do sensor ultrassônico;
- **D8**: sinal *echo* do sensor ultrassônico;
- **D10, D11, D12 e D13**: CS, MOSI, MISO e SCK, respectivamente, responsáveis pela comunicação SPI com o *Raspberry Pi*.

Sempre próximo aos pinos de alimentação de cada componente eletrônico foram posicionados capacitores de desacoplamento de 100nF. Antes de cada servo motor foram colocados dois capacitores em paralelo, um de 100nF e outro de 10μF, com o objetivo de filtrar espectros de frequência ruidosos, o primeiro como filtro passa-baixas e o segundo como filtro passa-altas. Outro capacitor que merece destaque é o que se observa no circuito auxiliar no valor de 1000μF, este precisava ter um valor alto de capacitância para garantir que o LED azul de alto brilho não sofresse com as oscilações de tensão da bateria Lipo. O ideal seria um de 10000μF, mas o valor mais alto encontrado para comprar foi o de 1000μF.

Um último detalhe a se comentar é a escolha dos resistores para os LED azul e vermelho. Um LED de alto brilho apresenta uma queda de tensão entre 3,0 e 3,2 volts, e para garantir que tenha uma longa vida é importante que a sua corrente seja inferior a 30mA. De acordo com a Lei de *Kirchhoff* das malhas, para um caso de 12,0V na bateria, e 3,2V no LED, teríamos

$$-12,0 + RI + 3,2 = 0, \tag{3.2}$$

o que resultaria em uma resistência R de no mínimo 300Ω. Levando isso e os valores comerciais em consideração, foi escolhido um valor de 470Ω que faz com que a corrente no LED seja de aproximadamente 18mA, um valor adequado para o problema proposto.

Concluída a parte de projeto dos esquemáticos iniciou-se a realização de testes para garantir

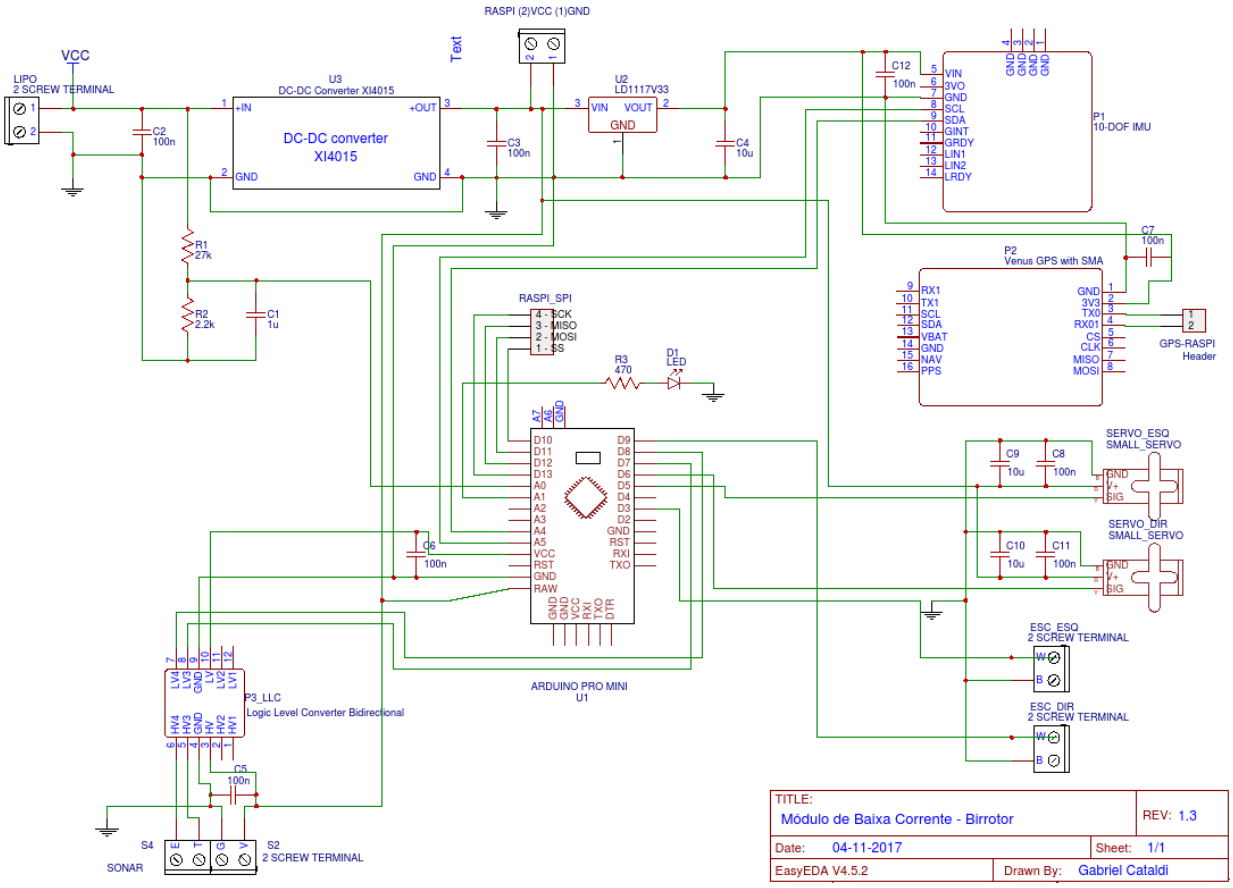


Figura 3.9: Esquemático detalhado do módulo de baixa corrente.

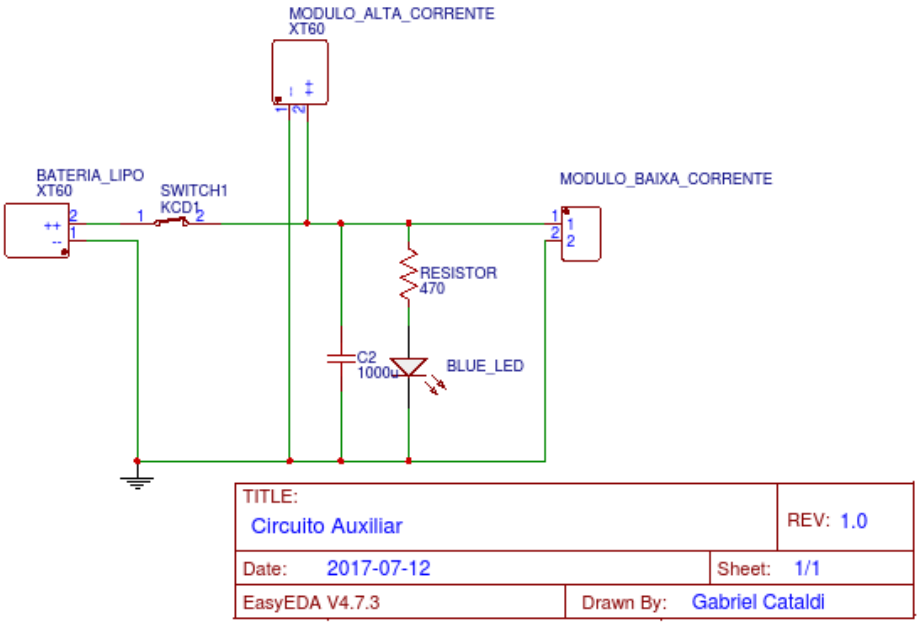


Figura 3.10: Esquemático do circuito auxiliar.

que o circuito projetado estivesse funcionando de modo apropriado. Utilizando uma *protoboard*[11] e uma fonte de alimentação de 12,0V para montar o circuito projetado e suas conexões, todos os componentes funcionaram perfeitamente, tanto os de baixa quanto os de alta corrente. Diante desses resultados positivos, pode-se passar para o próximo passo: a fabricação do circuito em uma placa.

3.2.4 Projeto e Fabricação da Placa

Primeiramente, é preciso transformar todos os esquemas desenhados em projetos de placas viáveis para fabricação e que se encaixem perfeitamente no veículo áereo projetado. Utilizando da mesma plataforma *online* apresentada na seção anterior, o *Easy EDA*, foram testados diferentes posicionamentos dos componentes em busca de uma boa otimização do espaço. Aspectos importantes no processo de *design* de uma placa de circuito estudados no livro *The Circuit Designer's Companion*, do autor Tim Williams[23], foram levados em consideração. Dessa forma, no processo de criação evitou-se trilhas que formassem ângulos de 90º, pois ângulos retos acabam atuando como paredes que prejudicam a transmissão de sinais. A Figura 3.11 mostra uma primeira proposta apresentada para o módulo de baixa corrente, com trilhas sempre com inclinações de 45º. As dimensões da placa ficaram de aproximadamente 99mm de comprimento e 84mm de largura, ou seja, seu tamanho ficaria perfeito para ser acoplada na base do veículo aéreo birrotor.

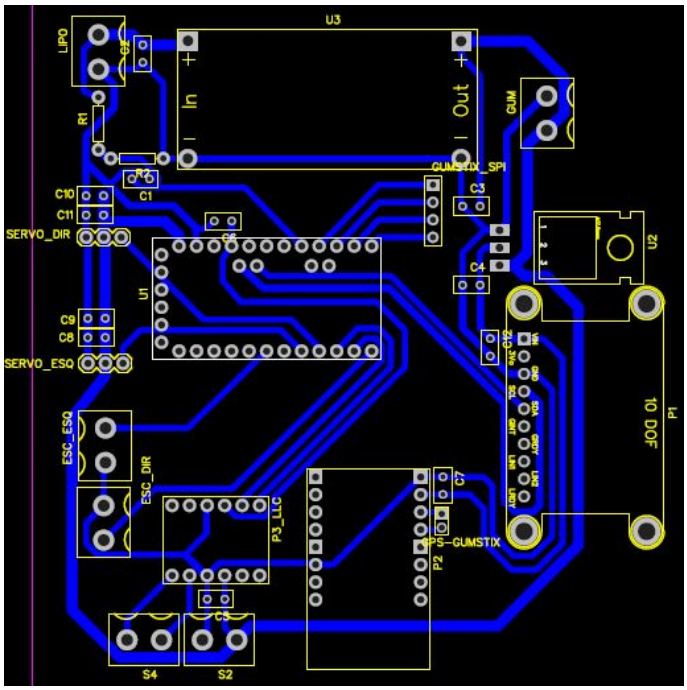


Figura 3.11: Primeira proposta de placa para o módulo de baixa corrente.

Lendo um pouco mais os conceitos abordados por Tim Williams em [23], uma segunda proposta foi trabalhada. Trilhas que levariam mais corrente precisariam de espessuras maiores que

[11] *Protoboard*: placa de ensaio ou matriz de contato cheia de furos e conexões condutoras para a realização de testes de circuitos elétricos

garantissem com folga que os componentes recebessem a quantidade necessária. Desse modo, três medidas de largura foram adotadas: 1,5mm para as de maior corrente, 1,0mm para médias, e 0,8mm para trilhas de baixa corrente. Além disso, uma abordagem utilizando plano de terra se mostrou mais interessante. A forma como um sinal retorna ao terra do sistema é muito importante, visto que quando ele toma um caminho mais longo pode gerar uma corrente de retorno, e assim, uma tensão ruidosa causada pela resistência das trilhas. Também devido ao comprimento das trilhas, existe a possibilidade de gerar uma indutância decorrente das malhas criadas. Portanto, a utilização de um plano de terra bem implementado pode acabar com essa corrente de retorno, eliminar ruídos na placa, e evitar indutâncias não previstas no projeto inicial. Ao considerar todos esses aspectos, chegou-se à solução apresentada na Figura 3.12, que foi julgada como sendo a melhor opção a ser fabricada. Não só pelos aspectos citados, mas também por conta do processo de fabricação que seria utilizado. Recentemente, o pesquisador do LARA, Miguel Eduardo Gutierrez, conseguiu deixar em perfeito funcionamento uma máquina fresadora CNC[12] para a fabricação de placas de circuito através de usinagem. Diante disso, a segunda proposta também era mais viável do ponto de vista de fabricação, porque com o plano de terra haveria um menor gasto de tempo de usinagem, um menor desgaste da ferramenta de fresa utilizada, e um menor gasto de tempo da equipe do laboratório. Vale ressaltar aqui a ajuda do Miguel Gutierrez e do Breno Mendes, sempre à disposição para auxiliar no uso da máquina.

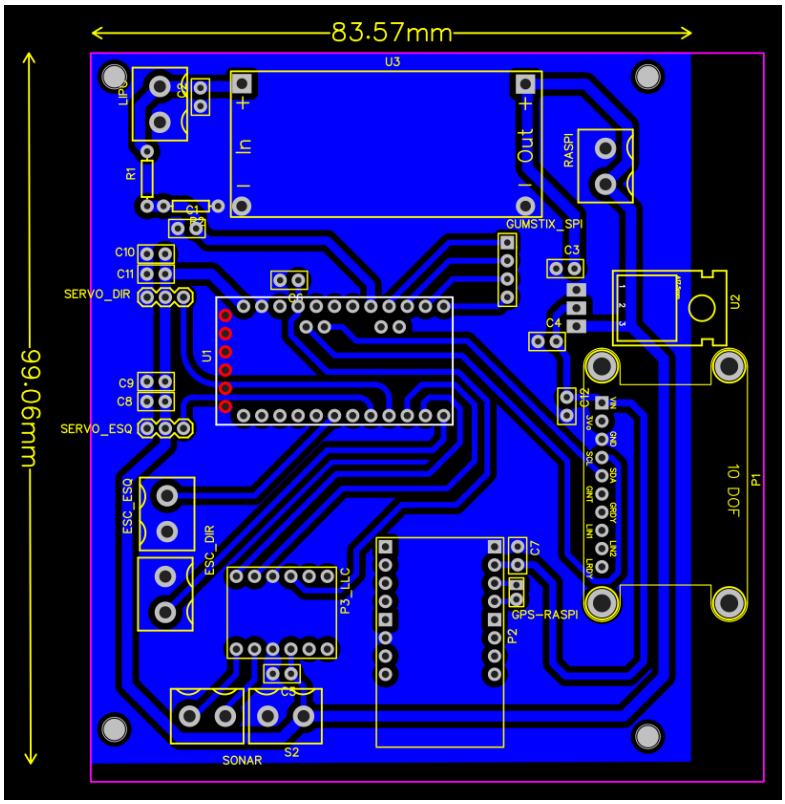


Figura 3.12: Segunda proposta de placa para o módulo de baixa corrente utilizando plano de terra.

O projeto da placa do circuito auxiliar também foi feito seguindo os mesmos princípios vistos

[12]CNC: do inglês *Computer Numeric Control*

em [23] e pode ser visualizado na Figura 3.13. As trilhas maiores possuem largura de 3,0mm para garantir que a corrente flua com folga da bateria para os dois módulos, principalmente no de alta corrente. Os conectores maiores são do tipo XT60, que encaixam com o utilizado pela bateria, que é do mesmo tipo.

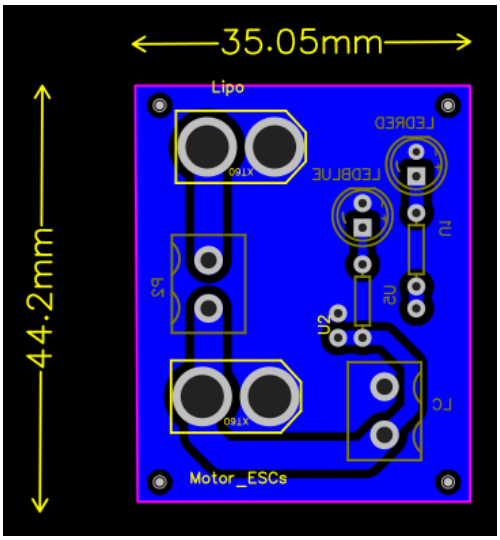


Figura 3.13: Placa para o circuito auxiliar utilizando plano de terra.

O processo de prototipagem de placas de circuito impresso, comumente chamadas de PCB[13], por meio de usinagem, mais especificamente fresagem, precisa antes da realização de alguns procedimentos. Primeiro, o projeto foi analisado e estruturado para ser fabricado de acordo com as limitações da máquina. Isso significa que os furos devem ter o diâmetro da broca disponível, e a distância entre as trilhas deve ter um valor mínimo de acordo com a espessura da fresa utilizada. Dito isso, todos os furos foram colocados com diâmetro de no mínimo 0,9mm e o mesmo valor foi usado também para definir a distância mínima de separação entre as trilhas. Os projetos apresentados nas Figuras 3.12 e 3.13 já estão com essas modificações incluídas.

Em seguida, a partir da plataforma de desenho utilizada, foram criados arquivos de fabricação no formato *Gerber*, que consiste em uma série de arquivos em um formato padrão universal que separa em camadas a estrutura da placa. Temos em cada arquivo uma camada que guarda algum tipo de informação sobre o projeto, por exemplo, sua camada inferior de cobre, ou então, o posicionamento dos furos, e assim por diante. Com esses arquivos em mãos, através de um programa chamado *FlatCam*[14], é possível organizar suas informações em um único arquivo, ajustar os parâmetros de usinagem como velocidade de avanço e tamanho das ferramentas e, a partir disso, criar um arquivo escrito em código G, que nada mais é do que uma linguagem de programação própria para dizer o que a máquina CNC deve executar. Neste último arquivo são encontradas todas as informações referentes aos caminhos que a fresa deve realizar para retirar o material de uma placa de cobre, a fim de obter a placa de circuito desejada. A partir do arquivo em código

[13] PCB: do inglês, *Printed Circuit Board*

[14] *FlatCam*: programa de prototipagem de placas de circuito impresso. Disponível em: <http://flatcam.org/>. Acesso em: 01 de novembro de 2017.

G, podemos então utilizar o programa *bCNC*[15] para verificar em simulação se o arquivo foi feito corretamente e realizar a interface diretamente com a máquina CNC para dar início ao processo de fabricação. A Figura 3.14 mostra o código G carregado no programa *bCNC*, mostrando o percurso que a ferramenta de fresa irá realizar durante o processo de usinagem. No total, duas placas foram usinadas: o módulo de baixa corrente, e o circuito auxiliar.

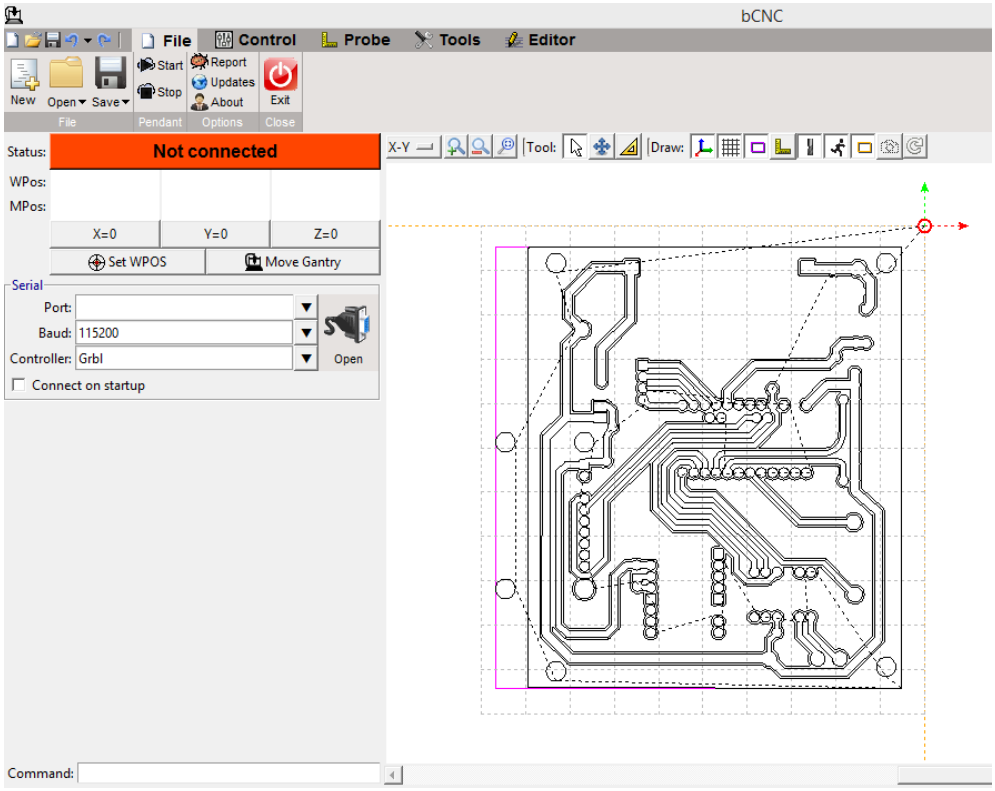


Figura 3.14: Código G do módulo de baixa corrente no programa *bCNC*.

3.2.5 Lista de Componentes

Como forma de documentação, nesta seção estão descritos todos os componentes mais importantes utilizados na construção do veículo aéreo e o modo como cada um atua no sistema birrotor. Alguns sofreram modificações ao longo do percurso, entretanto aqui estão relatados apenas aqueles que estão na versão final apresentada. Cada tabela a seguir representa um componente e suas respectivas especificações técnicas.

- *Joystick Freedom* 2.4GHz

Para controlar os atuadores do veículo aéreo birrotor é utilizado um *Joystick* da *Logitech*, como mostra a Tabela 3.2 com suas especificações. Ele possui um pequeno módulo sem fio que consegue se conectar via USB, o que lhe dá um alcance de até 6 metros de distância de operação. Para

[15] *bCNC*: programa que realiza a interface com a máquina CNC. Disponível em: <https://github.com/vlachoudis/bCNC/wiki>. Acesso em: 07 de novembro de 2017.

um VANT essa distância não é muita coisa, mas por enquanto é o suficiente para o objetivo do trabalho em questão.

Tabela 3.2: Especificações do *Joystick Logitech Freedom* 2,4GHz Sem Fio

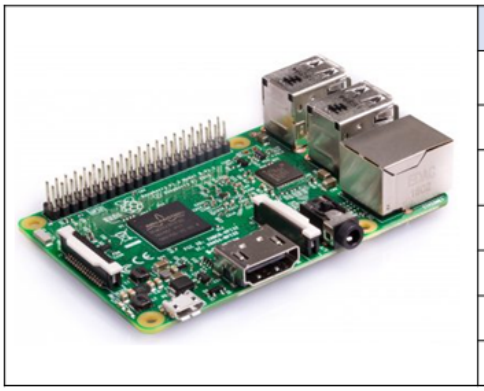
Características	Especificações
Conexão	Sem fio
Frequência de Operação	2,4GHz
Alcance	6 metros
Número de Eixos de Movimento	4 = X/Y/Z + *Throttle*
Número de Botões	10
Bateria	+50 horas com 3 pilhas AA
Peso	25g

Fonte: http://www.cybbay.com/store/logitech-freedom-2.4-ghz-cordless-joystick-963283-0403.html

- *Raspberry Pi 3 Modelo B*

O *Raspberry Pi 3* Modelo B possui instalado em seu cartão de memória um sistema operacional baseado em Linux chamado *Raspbian*. No sistema ele atua como o processador central que recebe informações dos periféricos e distribui os comandos necessários para o seu funcionamento. Ele é o cerébro do birrotor, recebe informações do *joystick*, envia comandos via comunicação SPI ao *Arduino Pro mini* e lê os dados por ele enviados, processando todas as informações essenciais ao sistema. A Tabela 3.3 mostra sua lista de especificações mais importantes.

Tabela 3.3: Especificações do *Raspberry Pi 3* Modelo B



Características	Descrição
Processador	Quad Core 1.2GHz Broadcom BCM2837 64bit
Memória RAM	1GB
Wireless	BCM43438 wireless LAN e Bluetooth Low Energy (BLE)
GPIO	40 pinos
USB	4 portas
Corrente Máxima	2,5A
Tensão	5V

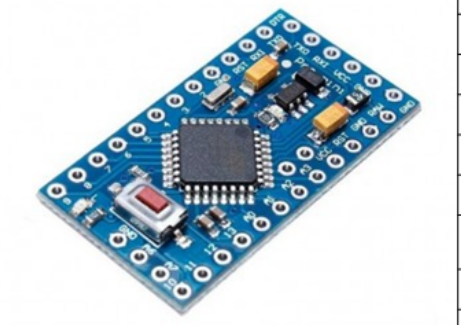
Fonte: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

- *Arduino Pro Mini*

O *Arduino Pro Mini* é utilizado como um processador auxiliar que faz a interface entre o *Raspberry Pi* e os sensores, os servo motores e os ESCs. Todo o trabalho pesado é feito pelo processador central, deixando para o *Arduino* apenas a função de obedecer os comandos que

recebe. Ele envia sinais PWM para os servo motores e para os ESCs e também se comunica via I2C com a unidade de medição inercial. Ele retorna os dados lidos ao *Raspberry* quando ordenado. A Tabela 3.4 mostra suas especificações.

Tabela 3.4: Especificações do *Arduino Pro Mini* 3,3V 8MHz



Características	Descrição
Microcontrolador	ATmega328P
Tensão de Operação	3,3V
Tensão de Alimentação	5,0 - 12,0V
Entradas e Saídas Digitais	14, das quais 6 podem ser PWM
Entradas Analógicas	8
Corrente DC por pino de I/O	40mA
Memória Flash	16kB, dos quais 2kB são usados pelo bootloader
Memória SRAM	1kB
EEPROM	512 bytes
Velocidade do Clock	8 MHz

Fonte: http://www.baudaeletronica.com.br/placa-pro-mini-atmega328p-5v-16mhz.html

- Servo Motores

Dois servo motores modelo HS-485HB, como descritos na Tabela 3.5, são utilizados, um posicionado em cada ponta da asa proporcionando os ângulos de inclinação nos rotores do veículo. Eles recebem comandos via PWM, nos quais a sua posição angular é diretamente proporcional à largura do pulso recebido. Quem envia os comandos aos servos é o *Arduino*, mas quem decide o que enviar é o *Raspberry*.

Tabela 3.5: Especificações do Servo Motor HS-485HB

Características	Descrição
Sinal	PWM
Corrente sem Carga (4,8V)	150mA
Torque sem Carga (4,8V)	4,82 kg-cm
Velocidade sem Carga (4,8V)	0.22sec/60°
Tensão de Operação	4,8 – 6,0V
Alcance Máximo do Sinal PWM	553-2425µs
Amplitude do Pulso	3,0 - 5,0V
Direção aumentando o sinal PWM	Sentido Horário
Material da Engrenagem	Carbonita
Peso	45g
Dimensões	39,88mm x 19,81mm x 37,85mm

Fonte: https://www.servocity.com/hs-485hb-servo

- Motores *Brushless*

Antes eram utilizados dois motores EMAX CF2822[16] que estavam disponíveis no Laboratório de Robótica Aérea. Entretanto, além desses motores estarem desgastados pelo tempo, seu empuxo máximo era capaz de levantar 710g; com os dois, ficaria algo em torno de 1420g. Como será detalhado na próxima seção, o peso do birrotor corresponderia a quase 80% do empuxo total. Houve a necessidade então de trocar os rotores utilizados para o modelo D2830/11 da *Turnigy*, que possui um empuxo máximo de 890g cada, somando um total de 1780g com os dois, diminuindo para quase 60% a relação peso/empuxo. Além disso, a qualidade dos motores *Turnigy* é visivelmente melhor. A perda da troca vem do lado do consumo de energia da bateria, que é maior por serem motores mais potentes. A Tabela 3.6 mostra suas características mais importantes.

Tabela 3.6: Especificações do Motor *Brushless* D2830/11

Características	Especificações
Modelo	D2830-11
Tensão	7,4 – 15,0V
KV (rpm/V)	1000
Empuxo Máximo	890g
Corrente Máxima	21A
ESC	30A
Potência Máxima	210W
Hélices Sugeridas (Bateria Sugerida)	8x4 (Lipo 4S) – 10x7 (Lipo 2S)
Peso	52g

Fonte: https://produto.mercadolivre.com.br/MLB-731983592-motor-brushless-turnigy-d283011-1000kv-aeromodelo-_JM

- *Electronic Speed Controller* – ESC

Com a troca dos dois rotores houve também a necessidade de trocar os ESCs. Antes eram utilizados modelos de 25A da EMAX[17], porém os novos motores consomem mais corrente. Consequentemente, a troca dos módulos de controle de velocidade também tinha que ser feita. Os novos ESCs além de possuirem uma capacidade maior de corrente de 30A, acabaram proporcionando também um ganho estrutural, visto que possuem um tamanho reduzido e um peso menor, com uma redução de 31g para 25g no peso de cada. A Tabela 3.7 possui um resumo de suas características. Cada ESC recebe comandos PWM através do *Arduino*, sendo a largura do pulso diretamente proporcional à velocidade de rotação dos motores sem escovas.

[16]Motor *Brushless* EMAX CF2822, disponível em: <http://www.baudaeletronica.com.br/motor-brushless-emax-cf2822-1200kv.html?gclid=Cj0KCQiA84rQBRDCARIsAPO8RFyfoMa6HLqwklGO01IC57Gax4ymejuUX-De0jjFqC9Lel7DU1apIvkaAvPXEALw_wcB>. Acesso em: 13 de novembro de 2017.

[17]ESC 25A EMAX, disponível em: <https://produto.mercadolivre.com.br/MLB-693497875-esc-emax-25a-bec-5v-2a-aeromodelo-_JM>. Acesso em: 13 de novembro de 2017.

Tabela 3.7: Especificações do *Electronic Speed Controller* 30A

Características	Descrição
Máxima Corrente de Saída	30A
Corrente de Pico	40A por 10 segundos
Tensão de Entrada	5,6 – 16,8V (2-4 células de lítio ou 5-12 células de NiCd/NiMH
BEC	2A/5V
Velocidade Máxima	210000rpm (2 polos BLM), 70000rpm (6 polos BLM), 35000 (12polos BLM)
Dimensões	45mm x 24mm x 11mm
Peso	25g

Fonte: https://produto.mercadolivre.com.br/MLB-858484766-esc-30a-brushless-bec-interno-2a-5v-aeromodelo-_JM

- Sensor Ultrassônico

O sensor ultrassônico possui a capacidade de medir distâncias em um alcance que varia de 2 centímetros até 4 metros. Como explicado anteriormente no Capítulo 2, seu funcionamento se baseia no tempo que uma onda sonora emitida em uma frequência de 40kHz demora para retornar. No veículo birrotor, esse sensor atua medindo a altura em relação ao solo. Por possuir um ângulo de efeito de apenas 15 graus, sua utilização durante um voo não é aconselhável, já que o robô em movimento estará sempre alterando sua inclinação em relação ao solo. Apesar disso, o sensor pode muito bem ser utilizado em uma situação de pouso e decolagem, e também quando o veículo estiver pairando (do inglês, *hovering*). A Tabela 3.8 mostra as suas características principais. Para realizar sua comunicação com o *Arduino* foi utilizado um conversor de nível lógico TTL[18], pois no sensor o nível lógico alto corresponde a 5,0V, enquanto no *Arduino* corresponde a 3,3V.

Tabela 3.8: Especificações do Sensor Ultrassônico HC-SR04

Características	Descrição
Tensão de Alimentação	5,0V
Corrente de Operação	15mA
Ângulo de Efeito	15º
Alcance	De 2cm até 4 metros
Precisão	3mm
Frequência de Operação	40Hz
Sinal de Entrada *Trigger*	10µs pulso TTL
Sinal de Saída *Echo*	Sinal TTL proporcional ao *trigger* e ao valor medido
Dimensões	45mm x 20mm x 15mm

Fonte: http://www.micropik.com/PDF/HCSR04.pdf

[18]Conversor de Nível Lógico TTL, disponível em: <https://produto.mercadolivre.com.br/MLB-725918895-conversor-de-nivel-logico-bidirecional-i2c-5v-33v-arduino-_JM>. Acesso em: 13 de novembro de 2017.

- IMU

A unidade de medição inercial utilizada é a 10 DOF (do inglês, *degrees of freedom*) da *Adafruit*. Em uma mesma placa possui acelerômetro e magnetômetro no módulo LSM303DLHC[24], girômetro no módulo L3GD20H[25] e um barômetro no módulo BMP180[26]. A comunicação da placa da *Adafruit* com o *Arduino* é feita através do protocolo I2C, utilizando os pinos SCL e SDA. No trabalho aqui desenvolvido, o sensor mais utilizado foi o girômetro, que possui a função de atuar como um giroscópio medindo o deslocamento angular em três eixos; em um veículo aéreo seria o deslocamento angular nos movimentos de arfagem, guinada e rolagem. A Tabela 3.9 mostra uma síntese das características mais importantes de cada módulo sensor e da placa como um todo.

Tabela 3.9: Especificações do *Adafruit 10-DOF IMU Breakout*

Características	Descrição
Tensão de Alimentação	2,2 – 3,6V
Temperatura de Operação	-40ºC a 85ºC
Escala do Girômetro	±245, ±500, ±2000 º/s
Escala Magnetômetro	±1,3 até ±8,1 gauss
Escala Acelerômetro	±2g, ±4g, ±8g, ±16g
Escala de Pressão	300 ate´1100hPa
Interface	I2C

Fonte: https://www.adafruit.com/product/1604

- Conversor DC/DC *Step-down*

Como explicado na seção 3.2.3, o conversor DC/DC LM2596 foi substituído pelo modelo XL4015 que possui uma bobina acoplada capaz de fornecer uma maior capacidade de corrente, podendo chegar a até 5A. Com esse novo conversor, cujas características estão retratadas na Tabela 3.10, foi possível realizar a alimentação de todos os componentes presentes no módulo de baixa corrente.

Tabela 3.10: Especificações do Conversor DC/DC *Step-down*

Características	Descrição
Tensão de Entrada (DC)	3,0 a 38,0V
Tensão de Saída (DC)	1,5 a 36,0V
Eficiência	Até 96%
Máxima Corrente de Saída	5A
Frequência de Comutação Fixa	180kHz
Dropout Mínimo	0,5V
Dimensões	49mm x 26mm x 19mm

Fonte: https://produto.mercadolivre.com.br/MLB-860244720-regulador-tenso-fonte-ajustavel-step-down-saida-12v-36v-5a-_JM

- Bateria

Infelizmente não havia disponível no laboratório nenhuma bateria, então foi adquirida uma Bateria Lipo da *ZIPPY Compact* com capacidade de 2200mAh e com tensão de 11,1V. Devido ao seu tamanho reduzido, o seu peso e sua alta capacidade, essa foi a melhor opção para alimentar todos os componentes do veículo áereo construído. A Tabela 3.11 informa suas especificações mais importantes.

Tabela 3.11: Especificações da Bateria Lipo *ZIPPY Compact* 2200mAh 3S 25C

Características	Descrição
Tensão	11,1v – 3 CÉLULAS
Descarga	25C Constante/ 35C de Pico
Capacidade	2200mAh
Plug de Balanceamento/Principal	JST-XH/XT60
Peso	179g
Dimensões	107mm x 21mm x 34mm

Fonte: https://hobbyking.com/pt_pt/zippy-compact-2200mah-3s-25c-lipo-pack.html

- GPS

O módulo GPS foi testado e está funcionando perfeitamente, porém sua integração com o *Raspberry* não foi finalizada. A sua comunicação utiliza uma interface UART LVTTL, utilizando seus pinos TX e RX, para transmitir e receber informações. Sua implementação via *software* e utilizando os pinos disponíveis no *Raspberry* é fácil de ser realizada, e deve ser retomada como um dos próximos passos de continuação do projeto. As características do GPS estão na Tabela 3.12. Para funcionar adequadamente, o GPS precisa estar conectado com uma antena, a utilizada é um modelo simples de antena de GPS automotivo[19].

Tabela 3.12: Especificações do GPS *Sparkfun Venus* com Conector SMA

Características	Descrição
Tensão de Alimentação	2,7 – 3,6V
Taxa de *Update*	Até 20Hz
Sensitividade	-148dBm, Partida Fria -165dBm, Rastreamento
Acurácia	Posição: 2,5 metros Velocidade: 0,1m/s Tempo: 60ns
Dinâmica	4G
Tempo de Reaquisição	Menor que 1s
Limites de Operação	Altitude: 18000m Velocidade: 515m/s

Fonte: https://www.sparkfun.com/products/11058

[19] Antena GPS, disponível em: <https://produto.mercadolivre.com.br/MLB-808182957-antena-gps-automotiva-_JM>

3.2.6 Projeto Mecânico

Um primeiro protótipo de um veículo aéreo com dois rotores articulados foi desenvolvido por Davi e Mickael no trabalho [5]. A Figura 3.15 mosta uma foto do VANT construído. Os autores, na página 85, destacam alguns pontos na estrutura que precisavam ser revistos futuramente:

- Um servomecanismo mais robusto, enxuto e confiável;
- Uma melhor forma de fixação dos ESCs;
- Parafusos que se afrouxam devido à vibrações excessivas na estrutura;
- A falta de um modelo 3D do sistema.

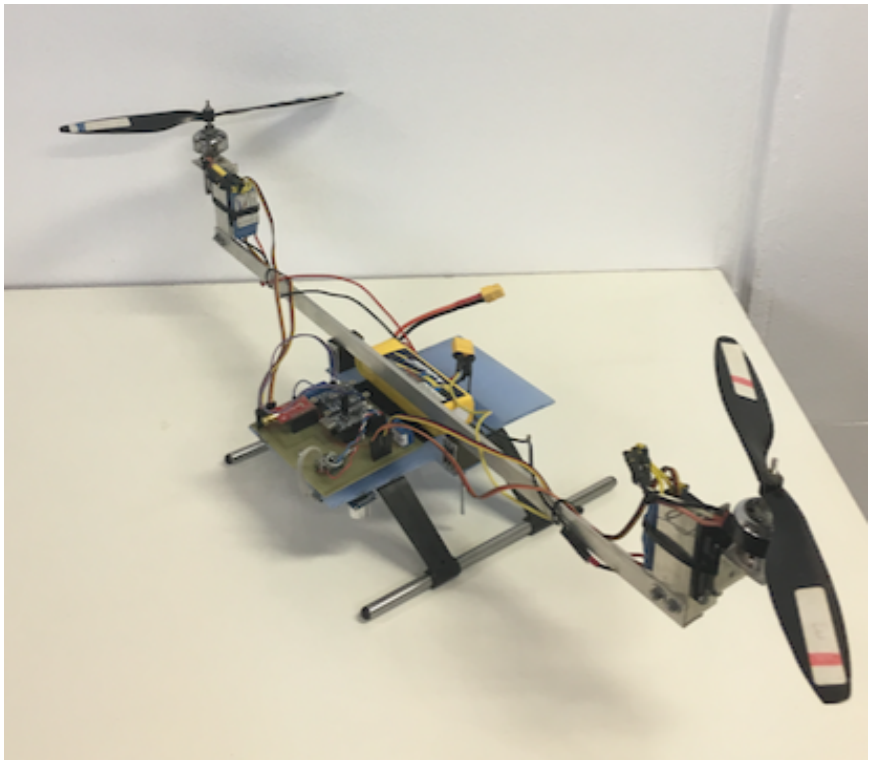


Figura 3.15: Foto em perspectiva do veículo áereo construído por Davi e Mickael (Fonte: foto de Davi e Mickael retirada de [5]).

A criação de modelos 3D é muito importante no desenvolvimento de qualquer projeto de engenharia, pois por meio deles é possível transmitir sua ideia, analisar suas características e compreender melhor o funcionamento daquilo com que se está trabalhando. Partindo disso, o primeiro passo do projeto mecânico foi realizar a medição de cada peça estrutural e criar um primeiro modelo 3D utilizando o programa de modelagem *SolidWorks*[20]. O resultado pode ser visto na Figura 3.16. Atividades como a desconstrução, a medição de cada peça, a remontagem, a criação de um modelo e a sua análise no computador acrescentaram bastante conhecimento sobre as necessidades do sistema como um todo.

[20] *SolidWorks*: disponível em <http://www.solidworksbrasil.com.br/>

Todos os pontos citados se confirmaram e outros foram observados. De fato, o posicionamento dos ESCs não estava adequado, principalmente por causa da disposição dos fios que estavam muito perto das hélices. Os fios dos motores *brushless* também estavam atrapalhando e criando uma resistência à movimentação do servomecanismo; durante a rotação dos servos eles ficavam se arrastando na estrutura de metal, o que poderia acarretar um possível rompimento, comprometendo sua integridade. O problema na fixação das peças também foi confirmado; não era o suficiente para aguentar as vibrações do sistema, levando os parafusos a se soltarem. Peças de metal estruturais não são muito utilizadas em projeto de veículos aéreos não tripulados, principalmente pelo peso. No caso, o eixo principal e o suporte dos servos utilizavam placas de alumínio em formato L, e além de serem de metal, seu formato assimétrico também contribui para o deslocamento do centro de massa do sistema.

Deslocamento este, acentuado pela disposição dos circuitos na placa de acrílico azul que serve como base. O peso do veículo aéreo era de 922g, entretanto esse peso foi medido sem o processador principal, sem a parte de potência, sem o circuito auxiliar de distribuição de energia e sem o módulo USB do *joystick*. Somando todas essas peças seu peso aumentaria muito e ficaria em condições impróprias de levantar voo, principalmente com os motores antigos que eram os CF2822 da EMAX capazes de um empuxo total de apenas 1420g.

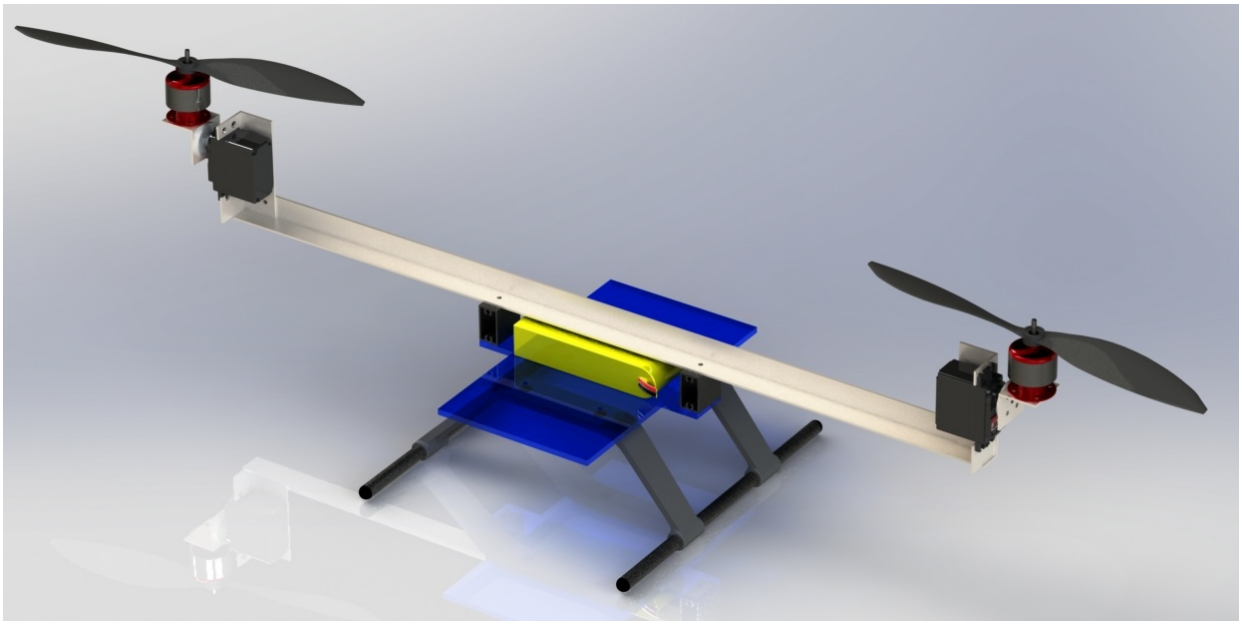


Figura 3.16: Modelo 3D do primeiro protótipo.

Diante de todos os argumentos citados, um novo projeto era necessário e alguns requisitos foram definidos. Para tentar garantir que o centro de massa fique o mais próximo possível do centro da estrutura, o posicionamento dos componentes seriam concentrados na região central; isso inclui a bateria, todos os circuitos e o *Raspberry*. Um canalizador para os fios que ligam os rotores aos ESCs era necessário para prendê-los e impedí-los de atrapalharem os servomecanismos. Um encaixe para o sensor ultrassônico na parte inferior, assim como um encaixe para a chave de liga/desliga e para os LEDs de alto brilho foram incluídos também como requisitos. O *design* da

estrutura foi repensado e a Figura 3.17 apresenta um primeiro conceito que serviu de referência para uma nova modelagem, em que apenas o trem de pouso seria reaproveitado de toda a parte estrutural.

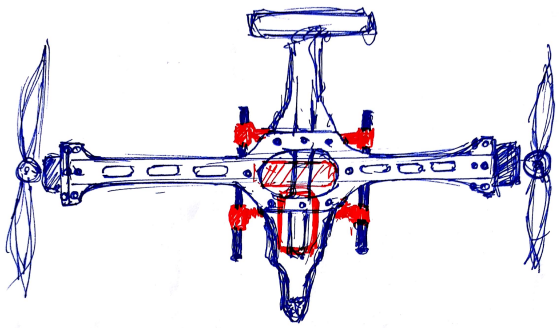


Figura 3.17: Conceito desenhado para um segundo protótipo.

A Figura 3.18 apresenta o novo modelo 3D criado utilizando o programa *SolidWorks*. Já na Figura 3.19, podemos observar alguns dos detalhes desse novo modelo enumerados para serem explicados individualmente. São eles:

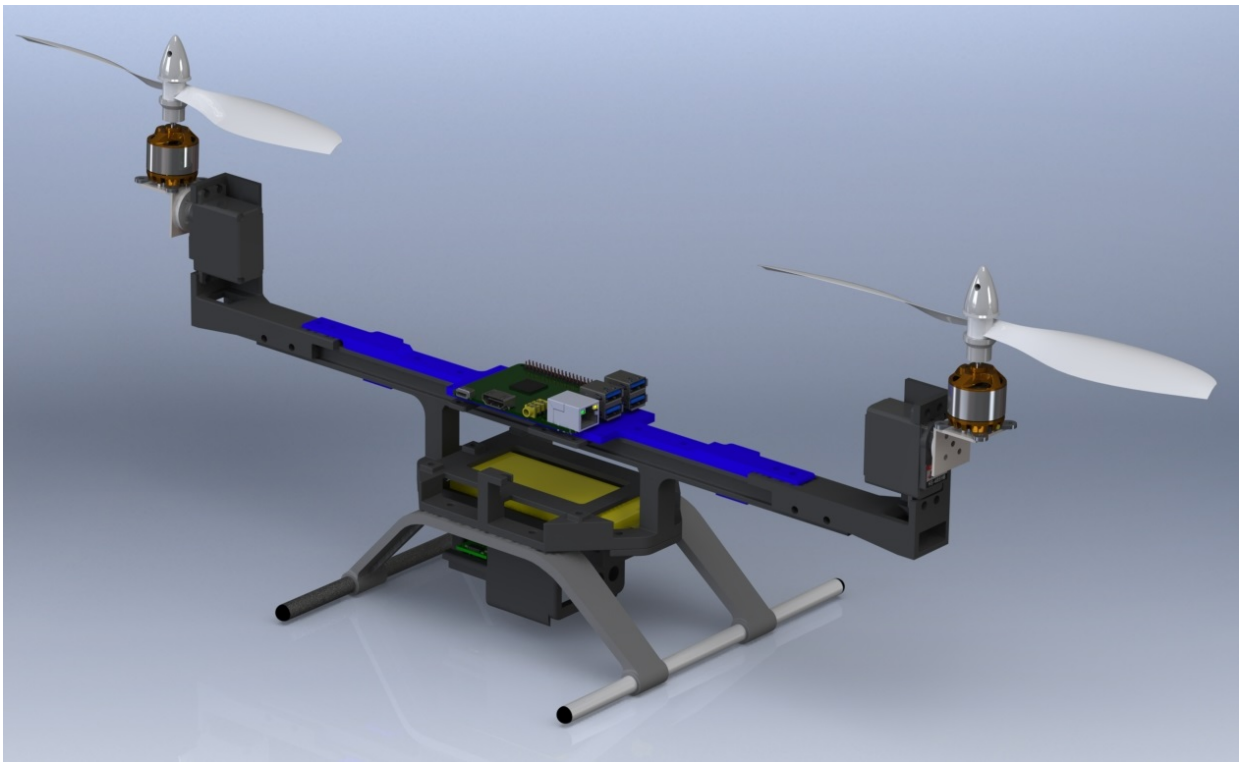


Figura 3.18: Modelo 3D novo do segundo protótipo.

1. Área de encaixe para a chave de liga/desliga e para os LEDs, um azul para indicar se o sistema está ligado e outro vermelho que indica se a bateria está fraca;

2. Espaços para a passagem de fios foram incluídos em diversos pontos chave;

3. Um suporte para a bateria Lipo foi adicionado, podendo ser desacoplado do resto da estrutura para sua remoção;

4. Os fios que conectam o ESC ao motor *brushless* agora passam por uma pequena cavidade abaixo do servo, impedindo que eles atrapalhem o servomecanismo;

5. Um suporte para fixar o sensor ultrassônico foi projetado para que ele fique embaixo do veículo;

6. Uma pequena torre foi incluída para prender a IMU e garantir que ela fique bem firme. Assegurar que ela fique bem presa é essencial para não atrapalhar as medições dos seus sensores imbutidos;

7. O eixo principal do birrotor sofreu uma redução no seu comprimento, para diminuir o peso, e para uma maior estabilidade do veículo.

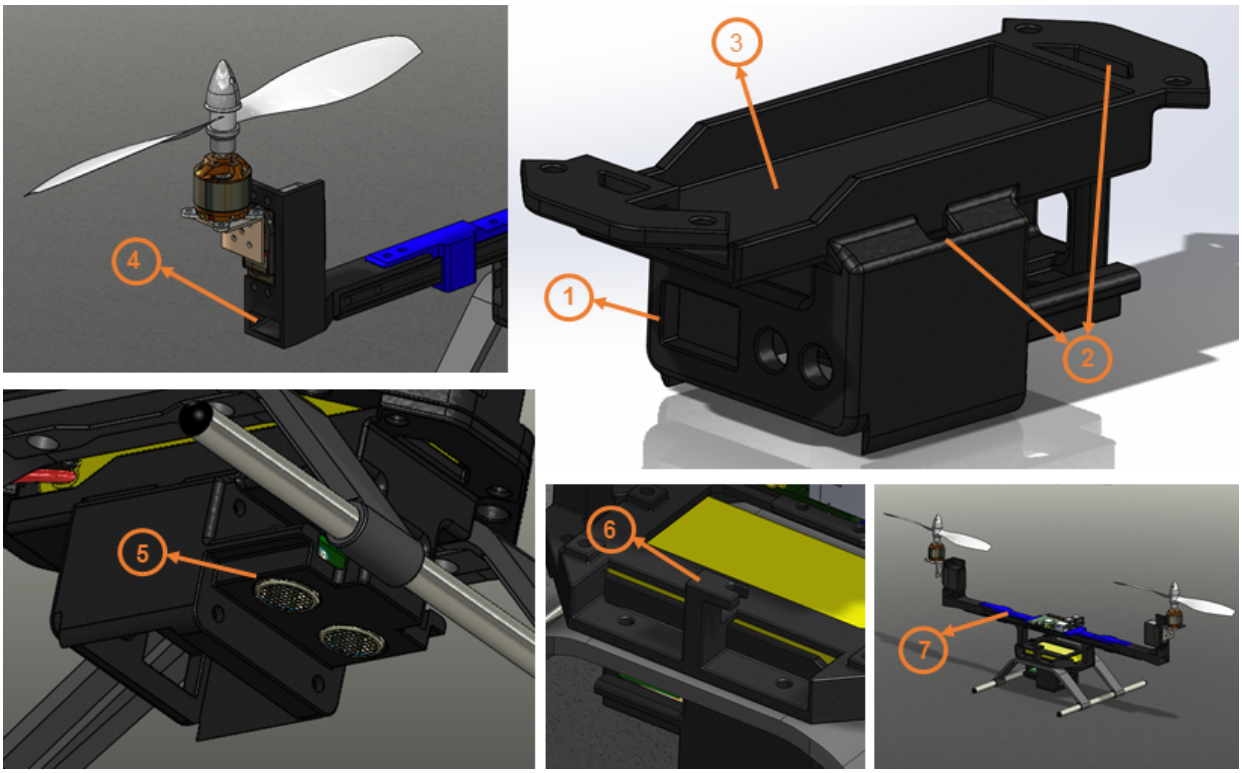


Figura 3.19: Detalhes do segundo protótipo.

Um dos objetivos estruturais era a redistribuição do peso, de modo que o centro de massa ficasse localizado o mais próximo possível do centro da estrutura. A Figura 3.20 mostra o centro de massa em diferentes vistas do modelo obtido através do programa *SolidWorks*, mais especificamente nas vistas frontal, superior e lateral direita. Na construção de veículos áereos é bastante utilizado fibra de carbono, isso por causa de sua elevada resistência e por ser muito leve, sua desvantagem vem no preço que o torna inviável no caso de um projeto de graduação. Diante disso, optou-se pela utilização de técnicas de fabricação por meio de impressoras 3D.

Todas as peças do novo modelo foram impressas utilizando um material chamado ABS, que vem de *Acrilonitrila Butadieno Estireno*, um polímero bastante utilizado nesse tipo de processo de fabricação e que possui uma boa combinação de propriedades mecânicas, elétricas, térmicas e químicas. A *Acrilonitrila* é responsável pela resistência química, o *Butadieno* pela resistência ao impacto e o *Estireno* pela rigidez e a facilidade de processamento [27]. Todas as peças foram impressas com apenas 20% de preenchimento interno, isso devido ao custo de fabricação, mas o ideal seria que todas fossem impressas com 100% de preenchimento para garantir a resistência e a rigidez necessárias para o veículo birrotor. Apenas os suportes dos servomecanismos na ponta de cada asa foram impressos com total preenchimento. A Figura 3.21 apresenta as peças impressas em ABS, todas elas foram pintadas de preto e montadas perfeitamente.

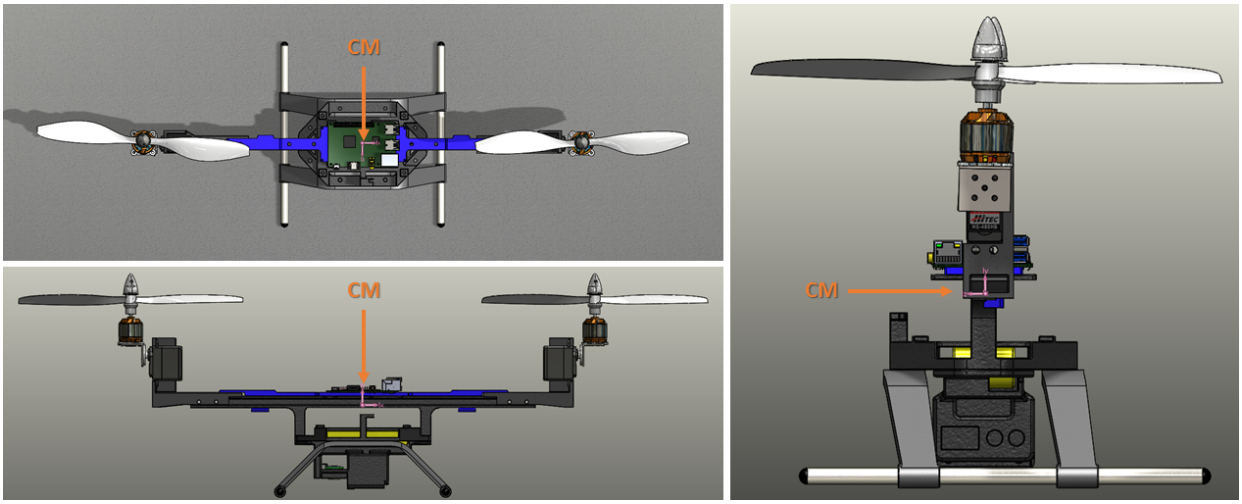


Figura 3.20: Centro de Massa (CM) do segundo protótipo.

Figura 3.21: Peças impressas em ABS à esquerda e montagem à direita.

Finalizando o projeto mecânico, foram compradas novas hélices para o modelo. As hélices antigas estavam coladas nos primeiros motores, elas eram muito velhas e o método de colagem não é o mais adequado de fixação, ainda mais porque os motores esquentam muito então sempre há o risco da cola se soltar em pleno voo. Duas hélices foram compradas com tamanho de 10x4,5 polegadas, uma com rotação para a direita (RHR: *Right Hand Rotation*) e outra com rotação para a esquerda (LHR: *Left Hand Rotation*). O sentido de rotação das hélices é muito importante para que o birrotor consiga o empuxo necessário para sair do chão e se mantenha estável[7]. A Figura 3.22 mostra o sentido correto adotado, com as duas hélices rodando para a região interna

do veículo em sentidos contrários, cancelando a ação do torque. Ambas as hélices foram presas no motor utilizando fixadores próprios como o mostrado pela Figura 3.23 que evitam acidentes e possuem uma maior garantia que as hélices não se soltarão.

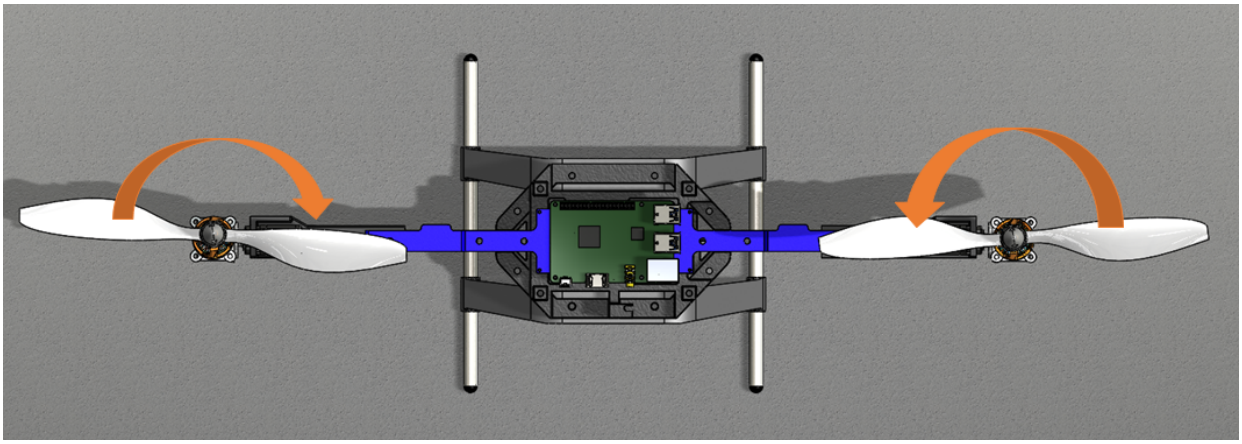


Figura 3.22: Sentido de rotação das hélices.

Figura 3.23: Hélices RHR e LHR à esquerda e fixadores à direita.

3.3 *Software*

3.3.1 Definição do Sistema

Antes de tudo, foi feito um estudo do que já havia sido implementado no trabalho anterior de Davi e Mickael[5], para se ter uma ideia de tudo aquilo que já estava desenvolvido e do que precisava ser trabalhado. Existe nesse processo uma dificuldade intrínseca ao desenvolvimento de *software* que é compreender códigos escritos por pessoas com diferentes hábitos e técnicas de programação. Dito isso, após todo esse processo, foi identificado que os algoritmos implementados foram feitos todos utilizando a IDE do *Arduino* com exceção da comunicação com o *joystick* que utilizou uma plataforma chamada *Processing* e um programa, ambos baseados em linguagem Java. Cada sensor possuía um código separado, ou seja, não havia a integração dos programas escritos para que todos pudessem ser executados em um mesmo código ao mesmo tempo em uma aplicação em tempo real. Assim como foi discutido na seção 3.2.1, toda a topologia precisou ser revista, e a do *software* principalmente. Em uma aplicação de robótica áerea, a parte de programação precisa

ser implementada de forma adequada para minimizar ao máximo a ocorrência de erros, pois um erro pode levar a um acidente e à consequente destruição do veículo birrotor.

Após a compreensão das características do sistema e das suas necessidades, iniciou-se o processo de projeto de um novo *software* baseado em Linux para funcionar no *Raspberry Pi*, este atuando como um sistema embarcado no veículo aéreo. Alguns pontos entraram em discussão: qual linguagem de programação seria utilizada, qual seria a melhor forma de implementar a interface com o *joystick*, e se seria utilizado o *FreeRTOS*[21], que consiste em uma plataforma para sistemas embarcados que permite organizar as tarefas e controlar seu escalonamento de acordo com a associação de prioridades. Diante da falta de experiência com essa ferramenta, acabou-se optando por não utilizá-la, embora fosse acrescentar bastante ao sistema. Em relação à interface com o *joystick*, a primeira opção foi de implementá-la através de uma plataforma como o ROS[22], que possui uma ampla gama de bibliotecas para diversos tipos de aplicações de robótica e uma interface para diferentes tipos de *joysticks*. Entretanto, o próprio Linux possui uma biblioteca disponível para realizar esse tipo de interface, a "*joystick.h*"[23], que foi escolhida para ser utilizada devido à sua praticidade, sua simples documentação e consequente facilidade de se trabalhar. A linguagem escolhida para a realização desse projeto foi o C++ por uma questão de experiência com a sua utilização.

Por meio de uma estratégia de desenvolvimento *top-down*[24], foi definida primeiro a estrutura do *software* em um nível mais alto. Apresentado na Figura 3.24, o projeto consiste na utilização de *threads*, que como explicado anteriormente, consistem em tarefas que são executadas de forma concorrente, ou seja, executadas ao mesmo tempo. O programa é divido em três *threads* que são inicializadas em uma rotina principal (*main*); cada *thread* será uma sequência de instruções em forma de laço, com requisitos temporais de execução. A rotina principal ficará presa em um laço infinito segurando cada *thread* até que um comando de finalizar o programa seja enviado pelo usuário através do *joystick*, o que encerrará todas as *threads* e por fim o *software*. Dessa maneira, temos as seguintes funções estabelecidas para cada *thread*:

- ***Thread Joystick***: faz a leitura dos comandos enviados pelo *joystick* e armazena os valores lidos em variáveis globais;
- ***Thread* Controle**: inicia a comunicação SPI com o *Arduino*, solicita a leitura dos sensores, calcula o controle e envia os comandos;
- ***Thread* GPS**: faz a leitura do GPS e armazena os valores em variáveis globais.

Seguindo a estratégia de desenvolvimento definida, agora será especificado mais a fundo cada detalhe do *software*. Em razão de uma melhor organização, o programa foi dividido em partes,

[21] FREERTOS: *Free Real Time Operating System*. Disponível em: <http://www.freertos.org>. Acesso em: 13 de novembro de 2017.

[22] ROS: *Robot Operating System*. Disponível em: <http://www.ros.org/>. Acesso em: 13 de novembro de 2017.

[23] Biblioteca do Linux *joystick.h*. Disponível em: <https://www.kernel.org/doc/Documentation/input/joystick-api.txt>. Acesso em: 13 de novembro de 2017.

[24] *Top-down*: do inglês, de cima para baixo. Abordagem em que cada nível é detalhado começando pelo nível mais alto até o nível mais baixo

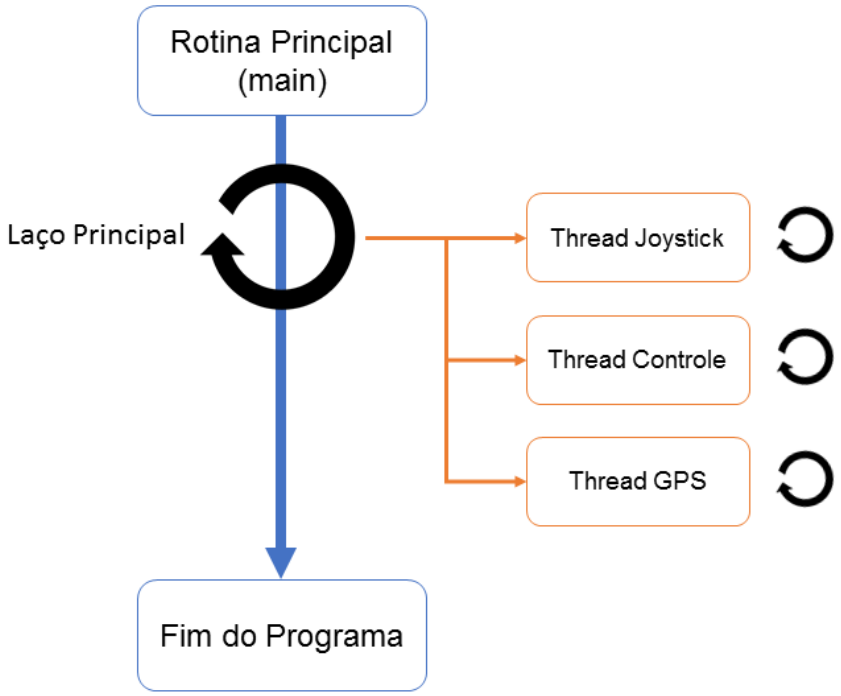


Figura 3.24: Estrutura do *software* em alto nível.

ou seja, em diferentes arquivos de código-fonte e seus respectivos cabeçalhos, mais conhecidos como *header files*. Cada código-fonte e seu respectivo *header* agrupam funções relacionadas, que compartilham de alguma característica em comum. Essa estratégia permite que códigos sejam reutilizados em outras aplicações, e concede uma maior praticidade na hora de analisá-los em decorrência de algum problema ou erro. O programa desenvolvido foi dividido então da seguinte forma:

- **main.cpp**: rotina principal do programa, responsável por manter o programa executando e por inicializar as *threads* referentes ao *joystick*, ao controle e ao GPS;
- **Joy.cpp, Joy.h**: conjunto de funções e definições que se referem ao manuseio do *joystick*;
- **Comunicacao.cpp, Comunicacao.h**: conjunto de funções e definições relacionadas à comunicação SPI entre o *Raspberry Pi* e o *Arduino*;
- **Datalogger.cpp, Datalogger.h**: conjunto de funções e definições que servem para realizar a interface com os códigos desenvolvidos pelo professor Geovany Araújo Borges na biblioteca *gdatalogger*. Esta permite que as variáveis e os dados obtidos pelo *software* sejam armazenados em um arquivo compatível com o MatLab[25], facilitando posterior análise das informações obtidas.

Cada parte da arquitetura do sistema, como elas se relacionam e suas respectivas funções implementadas foram detalhadas na Figura 3.25. No canto direito da imagem é retratado também

[25]MatLab: *Matrix Laboratory*. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 15 de novembro de 2017.

o código-fonte escrito para o *Arduino* e sua relação com o resto do programa, que por meio da comunicação SPI com o *Raspberry* recebe comandos e envia as informações obtidas.

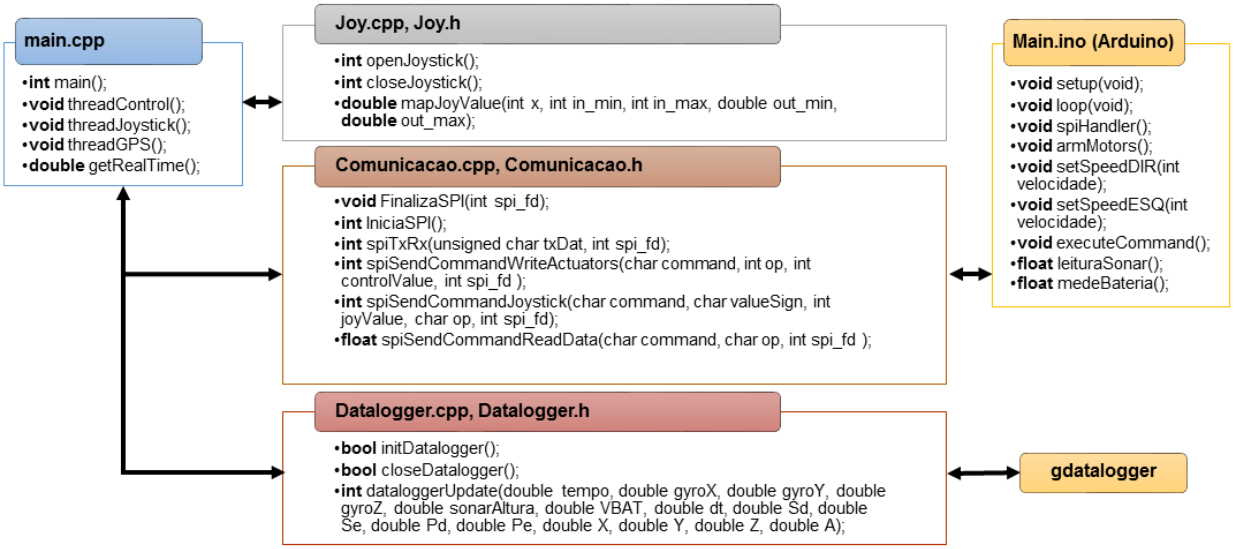


Figura 3.25: Estrutura do *software* em baixo nível.

3.3.2 *Joystick*

A leitura do *joystick* é realizada utilizando a API própria disponível no Linux, e os dados são lidos da mesma forma que se leem as informações de um arquivo qualquer. Assim, temos uma função que abre o arquivo de informações para leitura em modo *non-blocking* (do inglês, sem bloquear), que garante que o programa não seja bloqueado quando não tiver informações à serem lidas. E de modo análogo, temos uma função para fechar o arquivo quando o programa for encerrado. Os valores que o dispositivo envia variam de acordo com os diferentes eixos de movimento disponíveis ou então com os diferentes botões pressionados. O modelo utilizado da *Logitech* possui os três eixos para movimentação X, Y e Z, um mecanismo de controle deslizante utilizado para controlar a velocidade dos motores (chamado de A), e 10 botões distribuídos. Os botões possuem uma resposta binária, caso sejam pressionados mandam valor 1, caso não, mandam um valor 0. Já no caso dos eixos, os valores enviados são inteiros que variam em uma faixa de -32767 até 32767, com exceção do eixo Z, que possui um alcance inferior variando de -676 até -32767. Para uma melhor utilização no desenvolvimento da parte de controle, os valores foram mapeados utilizando a função mapJoyValue, da forma que todos assumissem o formato de pontos flutuantes e ficassem entre -1,0 e 1,0. A Tabela 3.13 mostra a representação dos eixos, o alcance de cada um deles e como foi feito seu mapeamento de valores. Como nomenclatura será usado o subscrito J para se referir aos valores mapeados. Já no caso dos botões, alguns foram escolhidos para funções específicas, são eles:

- **Botão 2**: Utilizado para encerrar a aplicação. Atua também como um botão de emergência de acesso rápido ao usuário, ao ser pressionado, imediatamente desliga os rotores e coloca os servos na posição inicial;

- **Botão 3**: Utilizado para ativar o modo de controle manual;
- **Botão 4**: Utilizado para ativar o modo de controle automático;

Tabela 3.13: Mapeamento de valores do *Joystick*

Eixos	Limites			
X		Esquerda	Centro	Direita
	Valores Lidos	-32767	0	32767
	X_J	-1,0	0	1,0
Y		Trás	Centro	Frente
	Valores Lidos	32767	0	-32767
	Y_J	-1,0	0	1,0
Z		Esquerda	Centro	Direita
	Valores Lidos	-32767	-18580	-676
	Z_J	1,0	0	-1,0
A (*Slider*)		Trás	Centro	Frente
	Valores Lidos	32767	0	-32767
	A_J	-1,0	0	1,0

3.3.3 Comunicação SPI

A comunicação SPI é um protocolo de alta velocidade que trabalha em modo *full-duplex*, ou seja, possui linhas separadas para enviar e receber dados, e pode fazê-los ao mesmo tempo. A comunicação entre o *Raspberry* e *Arduino* foi feita utilizando o protocolo *Spidev*[26] com a biblioteca "spidev.h", que já é disponibilizada no Kernel do Linux. O *Raspiberry* entra como o Mestre, enquanto o *Arduino* entra como o Escravo. Todas as características do protocolo são definidas pelo Mestre e é ele que gera o sinal de *Clock* que controla a comunicação. Várias possibilidades foram testadas até chegar nos seguintes parâmetros:

- **Tamanho da Palavra**: palavras com o tamanho de 8 *bits* são enviadas e recebidas em cada ciclo do relógio;
- **Taxa de Comunicação:** a frequência escolhida para comunicação entre os dois dispositivos foi de 2MHz;

Nos códigos "Comunicação.cpp" e "Comunicação.h" foram definidas rotinas que tratam o protocolo de comunicação SPI. Por meio de $IniciaSPI()$ o *driver* é inicializado e através do comando $ioctl()$[27] são ajustados seus parâmetros. Uma rotina para encerrar a comunicação também foi feita, a $FinalizaSPI()$. A função $spiTxRx()$ estabelece uma estrutura de dados da mesma forma que é definido pela biblioteca "spidev.h" e carrega os membros passando os dados e os parâmetros

[26]Spidev: protocolo de comunicação SPI disponível no Kernel do Linux. Informações em: <http://linux-sunxi.org/SPIdev>. Acesso em: 16 de novembro de 2017.

[27]ioctl(): é uma chamada do sistema para operações de entrada e saída em dispositivos específicos.

de configuração para o dispositivo através da *ioctl*(); desse modo, ela realiza a transmissão de apenas um *byte* e retorna um *byte* como resultado. A partir dessa rotina foram feitas funções específicas para cada situação que precisava ser tratada, são elas:

1. **spiSendCommandWriteActuators()**: essa rotina usa a função spiTxRx para enviar um comando formatado para o *Arduino* referente aos atuadores do sistema;
2. **spiSendCommandReadData()**: essa rotina usa a função spiTxRx para enviar um comando formatado que solicita a leitura de dados pelo *Arduino* e retorna o valor lido do tipo solicitado.

Cada uma dessas duas possui uma série de argumentos que definem o comando a ser enviado pelo Mestre ao Escravo. A partir de agora, apenas nesta seção do texto, a numeração apresentada na lista acima será utilizada para se referir à essas duas rotinas, isso para efeito de praticidade, evitando a repetição de seus nomes extensos. A Tabela 3.14 mostra a lista de comandos e como se dá a troca de mensagens entre os dois dispositivos, em azul as mensagens enviadas pelo Mestre e em vermelho as mensagens enviadas pelo Escravo. As funções 1 e 2 foram implementadas considerando um pacote fixo de informações que são transmitidas entre os dispositivos a cada chamada, sendo definido um pacote com tamanho de 8 *bytes*.

Explicando a tabela temos que, de um pacote de 8 bytes, o primeiro é sempre um comando chamado de *handshake*, que vem do inglês "aperto de mão", que é literalmente um cumprimento entre os dispositivos. O *Raspiberry* envia um caractere definido como "c" (destacado de azul) e o *Arduino* precisa responder com um caractere definido como "a" (destacado de vermelho). O Mestre carrega a informação nos registradores e quando o ciclo de relógio é ativado ele passa essa informação ao Escravo, este por sua vez carregará sua resposta nos registradores e apenas no próximo ciclo que a enviará. Se a resposta não ocorrer, o Mestre tenta de novo até conseguir e ele não enviará a sequência de instruções até que os dois estejam sincronizados.

Tabela 3.14: Tabela de comandos e troca de mensagens no protocolo SPI

<table>
<tr><th>Bytes</th><th>Handshake</th><th colspan="3">Girômetro</th><th colspan="3">Acelerômetro</th><th>Sonar</th><th colspan="2">Bateria</th><th colspan="4">Motores</th></tr>
<tr><td>[0]</td><td>c</td><td colspan="11"></td></tr>
<tr><td>[1]</td><td>a</td><td colspan="3">g</td><td colspan="3">a</td><td>s</td><td colspan="2">b</td><td colspan="4">m</td></tr>
<tr><td>[2]</td><td rowspan="8"></td><td>x</td><td>y</td><td>z</td><td>x</td><td>y</td><td>z</td><td>-</td><td>p</td><td>n</td><td rowspan="2">1</td><td rowspan="2">2</td><td rowspan="2">3</td><td rowspan="2">4</td></tr>
<tr><td>[3]</td><td colspan="3">0</td><td colspan="3">0</td><td>0</td><td colspan="2">0</td></tr>
<tr><td>[4]</td><td colspan="3">Sinal</td><td colspan="3">Sinal</td><td>Sinal</td><td colspan="2">Sinal</td><td rowspan="2">Pd</td><td rowspan="2">Pe</td><td rowspan="2">Sd</td><td rowspan="2">Se</td></tr>
<tr><td>[5]</td><td colspan="3" rowspan="3">Dado</td><td colspan="3" rowspan="3">Dado</td><td rowspan="3">Dado</td><td colspan="2" rowspan="3">Dado</td></tr>
<tr><td>[6]</td><td colspan="4">0</td></tr>
<tr><td>[7]</td><td colspan="4">0</td></tr>
</table>

Como visto na Tabela 3.14, cada componente tem um comando específico definido que o identifica. No caso dos sensores, existe um primeiro caractere que é o comando principal identificando qual tipo deve ser lido, e um segundo caractere que indica a operação de cada tipo. Por exemplo, no caso do girômetro, o primeiro caractere seria o comando "g", e o segundo caractere indicaria qual operação, ou seja, qual eixo que está sendo feita a medição, se é o "x", o "y" ou o "z". Do mesmo modo é feito para o acelerômetro, mudando no caso do sensor ultrassônico que não possui nenhuma operação, e também no caso da leitura da bateria, no qual o segundo caractere é utilizado para indicar se o alarme de bateria fraca deve ser acionado por "p", de positivo, ou não "n", de negativo. Após a identificação das ordens dadas pelo Mestre, o Escravo retorna os valores lidos e estes são armazenados nos próximos *bytes*, sempre com o primeiro reservado para identificar o sinal, identificando se o número enviado é positivo ou negativo. Para realizar o envio de números do tipo *float* foram feitos deslocamentos no número enviado, transformando ele para inteiro e depois de volta para ponto flutuante, tomando sempre o cuidado para não se perder informações.

Isso tudo explica o funcionamento da rotina 2. Na rotina 1, que se refere aos atuadores, acontece algo semelhante, o primeiro caractere indica que o comando é para os motores "m", depois dois *bytes* são utilizados para identificar a operação que direciona o comando para o motor apropriado, e mais dois bytes são utilizados para enviar o valor a ser escrito nos atuadores escolhidos. Algumas siglas para os motores foram utilizadas no desenvolvimento do *software* e serão usadas daqui pra frente, são elas:

- **Pd**: valor que indica a largura do pulso PWM no motor *brushless* direito (de 0 a 180);
- **Pe**: valor que indica a largura do pulso PWM no motor *brushless* esquerdo (de 0 a 180);
- **Sd**: valor que indica a posição angular do servo motor direito, dado pela largura do pulso PWM enviado (de 0 a 180 graus);
- **Se**: valor que indica a posição angular do servo motor esquerdo, dado pela largura do pulso PWM enviado (de 0 a 180 graus);

Por parte do *Arduino*, foi implementada uma rotina chamada spiHandler() que é responsável por tratar o protocolo de comunicação SPI recebendo as informações do *Raspberry* e interpretando o seu conteúdo, sempre mantendo o controle do tamanho dos pacotes de 8 *bytes* para não perder o sincronismo na troca de informações.

3.3.4 Controle

Um veículo aéreo não tripulado já possui uma complexidade de controle elevada. Quando se fala em um birrotor com rotores articulados essa complexidade aumenta, principalmente devido à sua configuração física. Como visto no Capítulo 1, muitos inventores e projetistas passaram anos de suas vidas buscando soluções para construir um veículo que agrupasse características VTOL e ao mesmo tempo características que permitissem uma velocidade e um alcance maiores. Dito isso,

nesta seção serão discutidos as propostas e os caminhos seguidos para controlar o veículo aéreo construído.

Primeiramente, é necessário apresentar com mais detalhes alguns pontos referentes ao novo VANT. Na Figura 3.26 temos a vista isométrica do veículo virado de frente que apresenta três sistemas de coordenadas adotados. O primeiro e principal, que se localiza no centro de massa e geométrico do veículo e é indicado pela cor laranja, retrata o sistema de coordenadas fixo no corpo. A partir dele que as medidas de orientação e posição da aeronave são obtidas. Os sistemas de coordenadas da cor amarela correspondem aos eixos fixos de cada servo motor designando as articulações dos rotores, com o eixo z indicando o sentido positivo de rotação de cada servo.

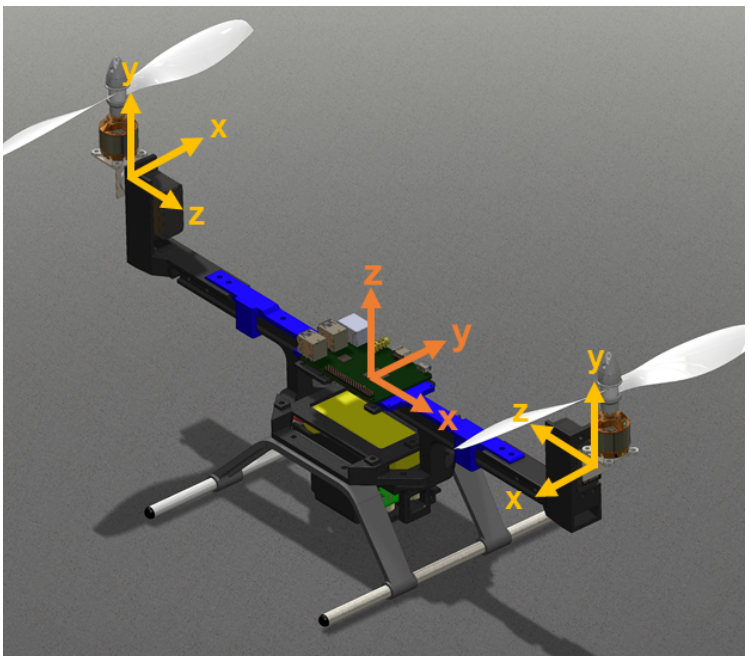


Figura 3.26: Sistema de coordenadas do veículo aéreo birrotor - Vista Isométrica.

É importante notar que os servos estão invertidos um em relação ao outro, como é possível observar também na Figura 3.27, que mostra, com um maior nível de detalhes, o sistema de referência adotado em cada um deles. Cada servo realiza uma movimentação de 0 a 180 graus, mas neste trabalho foram limitados entre 45 e 135 graus neste trabalho. A posição de 90 graus indica o centro, ou mais especificamente, indica a posição central utilizada como referência para decolagens, pousos e voos pairados. A posição angular é dada pelas variáveis Sd e Se que assumem valores entre 45 e 135 graus, enquanto a potência dos rotores é dada pelas variáveis Pd e Pe que assumem valores entre 0 e 180. A variação da posição angular em cada servo motor é medida a partir do centro (90^o) e toda vez que nos referirmos à ela utilizaremos a letra grega β.

Quando se fala em controle de um veículo aéreo não tripulado, se fala em controle de atitude, controle de altitude e controle de posição, assim como mostrado no diagrama simplificado da Figura 3.28. O controle de posição é o último a ser implementado, precisa que os outros dois estejam funcionando bem para que, por meio de um monitoramento via GPS e acelerômetro, seja possível calcular o posicionamento do veículo. O controle de altitude é feito aumentando ou

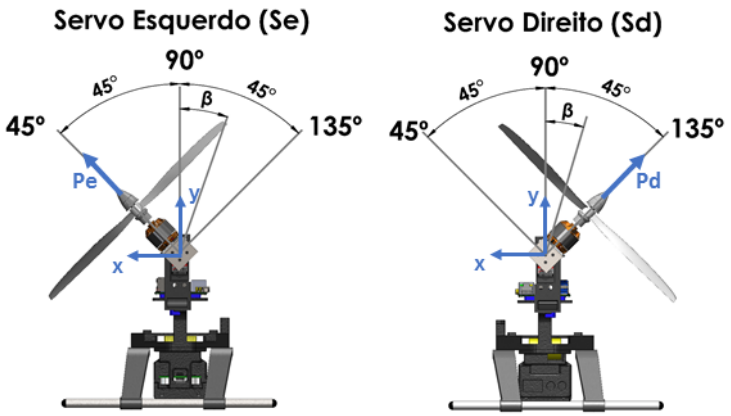


Figura 3.27: Vista lateral dos servo motores.

diminuindo a potência dos rotores, fazendo o veículo subir e descer. Seu monitoramento é feito através da união entre as informações do GPS e do sensor ultrassônico. Para o bom funcionamento dos dois é essencial um controle de atitude bem feito. Este último será o foco deste trabalho e por isso está destacado com a cor laranja na Figura 3.28.

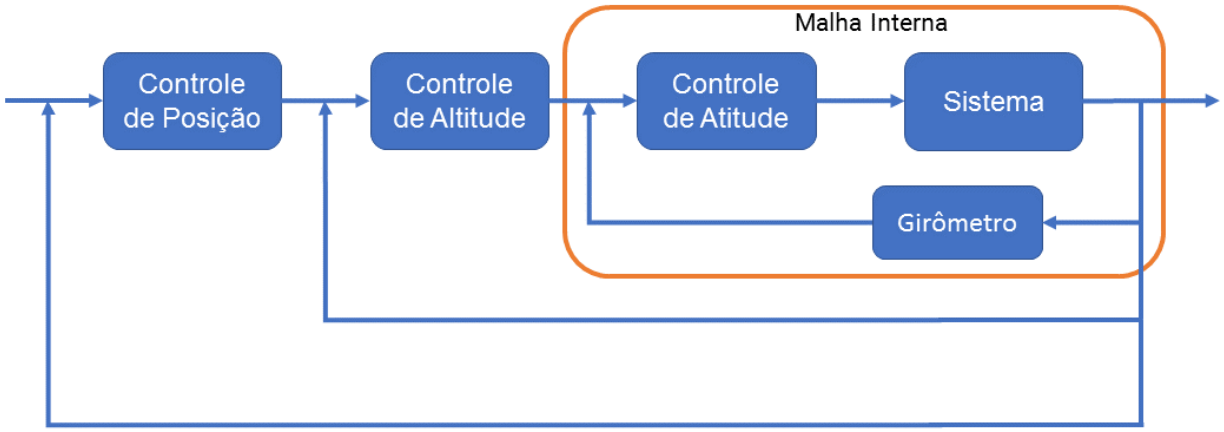


Figura 3.28: Arquitetura de controle simplificada de um VANT.

O controle de atitude é realizado por meio da medição das velocidades angulares em cada um dos movimentos de rotação representados pelos ângulos de Euler, [ψ, θ, ϕ], que descrevem, em uma forma mínima de representação, a orientação de um corpo rígido girante em relação ao seu sistema de coordenadas fixo estabelecido[28]. Esta representação angular é bastante utilizada em aplicações aeronáuticas e cada ângulo indica uma rotação em torno de algum eixo do sistema. A Figura 3.29 representa a dinâmica do veículo aéreo, mostrando os três tipos de movimento em seu sentido de rotação positivo utilizando as representações angulares de Euler definidas em cima do sistema de coordenadas fixo escolhido, são eles:

- **Guinada**: rotação em torno do eixo z definido, indicado pelo ângulo ψ;
- **Arfagem**: rotação em torno do eixo x definido, indicado pelo ângulo θ;

- **Rolagem**: rotação em torno do eixo y definido, indicado pelo ângulo ϕ.

Durante a arfagem, os rotores são movimentados simultaneamente para frente ou para trás, gerando um deslocamento no veículo para frente ou para trás, respectivamente, rotacionando o veículo em torno do eixo x. No caso da guinada, cada rotor é movimentado em sentidos opostos causando uma rotação em torno do eixo z. Já o controle da rolagem é feito variando a velocidade entre os rotores, ou seja, uma velocidade do direito maior que a velocidade do esquerdo fará o veículo rotacionar em torno do eixo y para a esquerda, e o contrário fará com que gire para a direita.

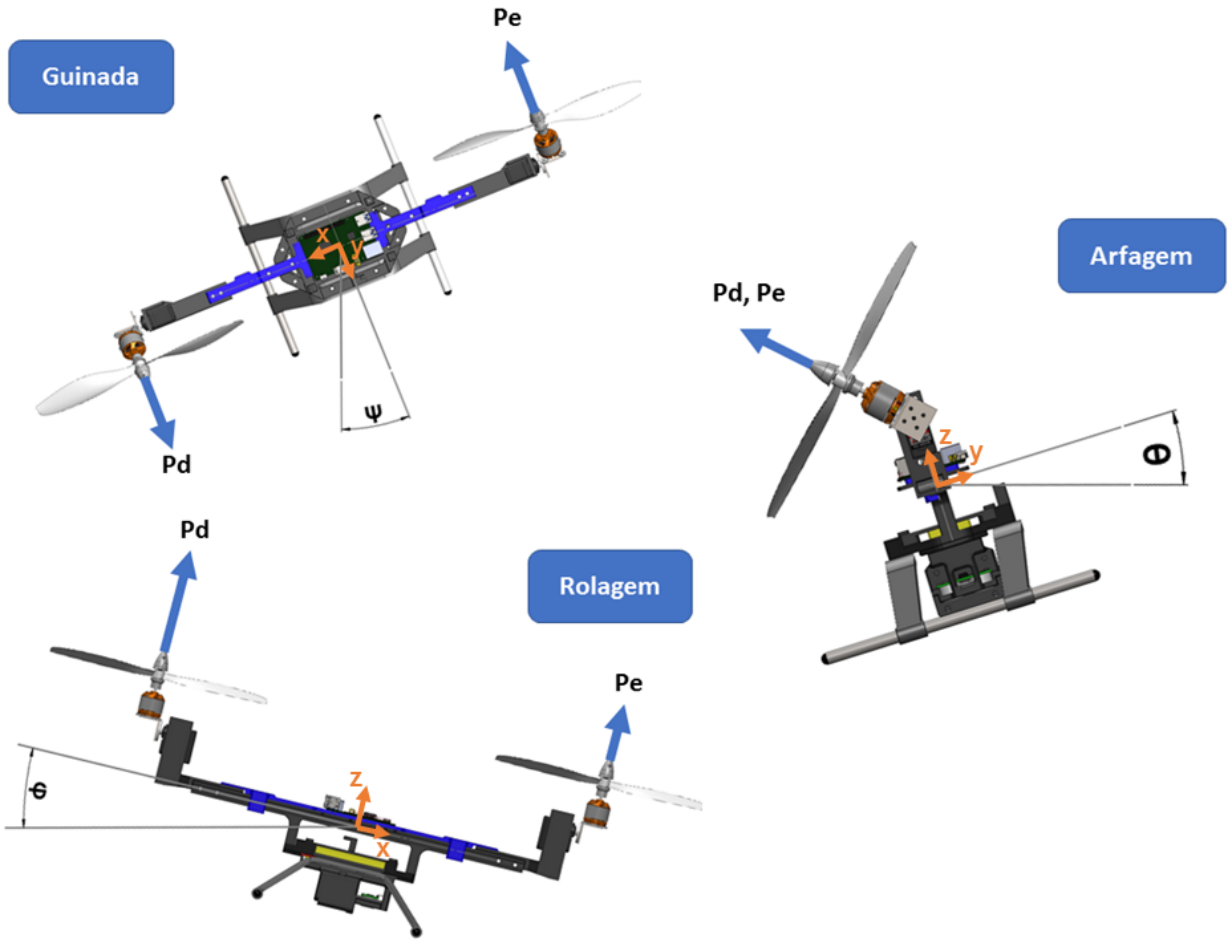


Figura 3.29: Movimentação do veículo aéreo birrotor.

Optou-se por uma abordagem mais direta e dois modos de controle foram implementados: um que chamaremos de Modo *Joystick* que fornece ao usuário através do *joystick* total domínio sobre o controle de atitude, de altitude e de posição do birrotor; e o outro que chamaremos de Modo de Estabilização que age como um piloto automático para garantir a estabilização da orientação do veículo áereo através do controle apenas de atitude. O Modo *Joystick* foi o primeiro a ser trabalhado e a sua abordagem inicial se deu relacionando os valores das posições dos servos com os comandos enviados pelo usuário através do dispositivo. Assim, era possível controlar em malha aberta os ângulos das articulações da aeronave diretamente com os valores de entrada recebidos,

que como apresentados na Tabela 3.13 da seção 3.3.2 flutuam entre -1,0 e 1,0 e dependem do eixo de movimentação escolhido: X, Y, Z ou A. Cabe destacar que esse sistema de coordenadas não é o mesmo que o do veículo; letras maiúsculas serão usadas para se referir aos eixos do *joystick* e letras minúsculas cursivas para os da aeronave. Dito isso, utilizando ainda a Tabela 3.13 como referência, foi definida a dinâmica do birrotor do seguinte modo:

- **Eixo X do *Joystick***: responsável pelo movimento de rolagem, rotação em torno do eixo y do sistema de coordenadas fixo do birrotor;
- **Eixo Y do *Joystick***: responsável pelo movimento de arfagem, rotação em torno do eixo x do sistema de coordenadas fixo do birrotor;
- **Eixo Z do *Joystick***: responsável pelo movimento de guinada, rotação em torno do eixo z do sistema de coordenadas fixo do birrotor;
- **Eixo A do *Joystick***: responsável pelo controle de altitude, aumentando e diminuindo a potência dos rotores.

Contudo, essa metodologia apresentava um problema. A resposta nos atuadores fica muito rápida, o que pode causar uma mudança de orientação muito brusca em uma premissa de voo, na qual os rotores estão operando em uma potência alta com propulsões elevadas. Nessas condições, uma leve inclinação é capaz de alterar completamente a orientação do veículo. Em razão disso, optou-se por implementar um controle em malha fechada do tipo Proporcional, que por meio do ajuste de Kp garantiu uma melhor interface homem-máquina. Diminuindo o valor do ganho proporcional do controlador, houve uma redução na sensibilidade do sistema e uma demora maior para a saída atingir o estado de referência desejado, propiciando uma sensação melhor ao controlar o veículo. A Figura 3.30 mostra o diagrama de blocos para o controle proporcional implementado, e seguindo o diagrama as posições angulares dos servos ficaram definidas como

$$Sd_t = Sd_{t-1} + K_p(Sd_{des} - Sd_{t-1}), \tag{3.3}$$

$$Se_t = Se_{t-1} + K_p(Se_{des} - Se_{t-1}), \tag{3.4}$$

indicando para o instante t o valor atual e para o instante $(t-1)$ o valor de saída obtido na última iteração no *software*. A posição desejada ou de referência (subscrito "des") é obtida pela relação da entrada do *joystick* em cada eixo com a posição angular correspondente. Cada movimento tem seus casos especiais determinados por meio de testes realizados no *joystick* e implementados via *software*. Da mesma forma, o controle proporcional da potência dos rotores ficou definido na forma de

$$Pd_t = Pd_{t-1} + K_p(Pd_{des} - Pd_{t-1}), \tag{3.5}$$

$$Pe_t = Pe_{t-1} + K_p(Pe_{des} - Pe_{t-1}), \tag{3.6}$$

sendo a potência desejada obtida pela entrada do *joystick* no eixo A.

Embora o controle de altitude ainda seja feito pela variação do eixo A, o Modo de Estabilização não utiliza a movimentação baseada no *joystick*. Ele é baseado na medição da velocidade angular

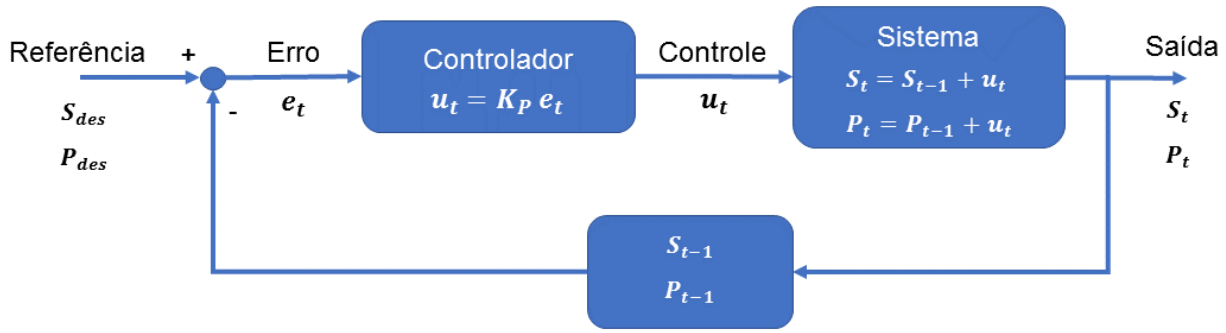


Figura 3.30: Diagrama de blocos do controle proporcional no Modo *Joystick* (Fonte: produzido pelo autor).

em cada um dos eixos fixos do VANT, que é feita através de um girômetro disponível na unidade de medição inercial. O objetivo principal dessa configuração é atingir uma posição de voo pairado e para isso, a proposta adotada foi a de utilizar um controle Proporcional-Derivativo(PD) para estabilizar o sistema. A Figura 3.31 mostra o diagrama de blocos que descreve seu funcionamento. Como visto anteriormente, os eixos de referência dos servos são invertidos, então é importante que assim como no Modo *Joystick* cada tipo de movimentação seja antes estudado individualmente.

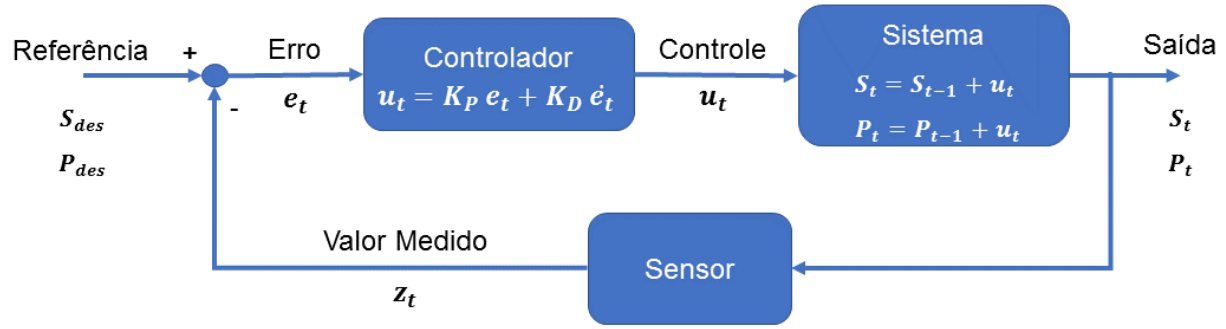


Figura 3.31: Diagrama de blocos do controle Proporcional-Derivativo no Modo de Estabilização.

Analisando o controle de posição dos servos, ambos precisam compensar a rotação na guinada e na arfagem se movendo no sentido contrário ao do movimento. Para isso, é preciso determinar as partes proporcionais e derivativas que constituirão a lei de controle do sistema. Analisando a guinada, tem-se que seus componentes derivativos são definidos como

$$Sd_t+ = K_{d1} \times (\dot{\psi}_{des} - \dot{\psi}_{t-1}), \tag{3.7}$$

$$Se_t+ = K_{d1} \times (\dot{\psi}_{des} - \dot{\psi}_{t-1}), \tag{3.8}$$

de modo que $\dot{\psi}$ é a velocidade angular em torno do eixo z e Kd_1 o seu ganho derivativo. De forma semelhante, se obtém os componentes derivativos do controle para a arfagem, definidos como

$$Sd_t+ = K_{d2} \times (\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{3.9}$$

$$Se_t+ = -K_{d2} \times (\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{3.10}$$

sendo $\dot{\theta}$ a velocidade angular em torno do eixo x e Kd_2 o seu ganho derivativo. Os sinais dos ganhos são invertidos justamente pelo fato do sistema de coordenadas de cada servo estarem também invertidos. E também tem-se

$$Sd_t+ = K_{p1} \times (Sd_{des} - Sd_{t-1}), \tag{3.11}$$

$$Se_t+ = K_{p1} \times (Se_{des} - Se_{t-1}), \tag{3.12}$$

que correspondem aos componentes proporcionais do controle de guinada e de arfagem. Quando se fala no controle de rolagem, se fala da mudança de velocidade dos rotores, então seus componentes derivativos são dados na forma

$$Pd_t+ = K_{d3} \times (\dot{\phi}_{des} - \dot{\phi}_{t-1}), \tag{3.13}$$

$$Pe_t+ = -K_{d3} \times (\dot{\phi}_{des} - \dot{\phi}_{t-1}), \tag{3.14}$$

nas quais $\dot{\phi}$ representa a velocidade angular em torno do eixo y e Kd_3 o seu ganho derivativo. E também a parte proporcional

$$Pd_t+ = K_{p2} \times (Pd_{des} - Pd_{t-1}), \tag{3.15}$$

$$Pe_t+ = K_{p2} \times (Pe_{des} - Pe_{t-1}), \tag{3.16}$$

nas quais Pd e Pe de referência serão dados pelo comando do *joystick*. Por último, somados todos os componentes, obtém-se as leis que definem o controle de orientação do veículo áereo birrotor, dadas pelas equações

$$Sd_t = Sd_{t-1} + K_{p1} \times (Sd_{des} - Sd_{t-1}) + K_{d1} \times (\dot{\psi}_{des} - \dot{\psi}_{t-1}) + K_{d2} \times (\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{3.17}$$

$$Se_t = Se_{t-1} + K_{p1} \times (Se_{des} - Se_{t-1}) + K_{d1} \times (\dot{\psi}_{des} - \dot{\psi}_{t-1}) - K_{d2} \times (\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{3.18}$$

$$Pd_t = Pd_{t-1} + K_{p2} \times (Pd_{des} - Pd_{t-1}) + K_{d3} \times (\dot{\phi}_{des} - \dot{\phi}_{t-1}), \tag{3.19}$$

$$Pe_t = Pe_{t-1} + K_{p2} \times (Pe_{des} - Pe_{t-1}) - K_{d3} \times (\dot{\phi}_{des} - \dot{\phi}_{t-1}), \tag{3.20}$$

sendo as duas primeiras responsáveis pelo controle da guinada e da arfagem, e as duas últimas responsáveis pelo controle da rolagem. Para o caso de estabilização do veículo aéreo, os valores desejados para as velocidades angulares seriam definidos como zero.

Um último detalhe precisava ainda ser considerado. A mudança na posição angular das articulações leva a uma decomposição do vetor da força dos rotores em uma componente na vertical e outra na horizontal. Diante disso, uma inclinação dos servos pode levar a perdas de altitudes não previstas já que apenas a componente vertical será responsável pela sua sustentação. A solução adotada foi considerar o ângulo β no controle de Pd e Pe de acordo com a seguinte atribuição implementada no programa

$$Pd_t = \frac{Pd_t}{cos(\frac{\beta}{2})}, \tag{3.21}$$

$$Pe_t = \frac{Pe_t}{cos(\frac{\beta}{2})}. \tag{3.22}$$

Uma primeira proposta usava apenas $\cos(\beta)$, mas o valor de Pd e Pe aumentava até 41% para uma variação angular de 45 graus. Um aumento desse tanto causou acidentes desastrosos. Portanto, por meio de testes com o veículo, o valor de $cos(\frac{\beta}{2})$ foi definido, definindo um aumento de até 8% na potência dos motores para o caso limite.

Para proteger o sistema foram inserido saturadores via *software* limitando as saídas para a posição dos servos e para a potência dos motores. Como já foi explicado anteriormente, os servos são limitados entre 45 e 135 graus. Já a potência dos rotores foi limitada entre 0 e 160, ou seja, seu limite máximo é de aproximadamente 89% da sua potência total. Mais do que isso não é necessário para a aplicação em questão, além de esquentar muito os componentes podendo até danificá-los.

3.3.5 GPS

A *thread* referente ao GPS infelizmente não chegou a ser implementada por falta de tempo, mas a estrutura do *software* permite facilmente que essa parte seja adicionada. Basta que seja feita a conexão do módulo GPS aos pinos do *Raspberry Pi* e a implementação da leitura através dos pinos Rx e Tx por meio de uma comunicação UART LVTTL.

Capítulo 4

Resultados

Neste capítulo serão apresentados todos os resultados obtidos nos projetos de *hardware* e de *software*, mostrando também os resultados do sistema completo funcionando, com os algoritmos de controle de atitude e estabilização implementados.

4.1 Projeto Eletrônico

Como resultado do projeto eletrônico desenvolvido, duas placas foram construídas: o módulo de baixa corrente e o circuito auxiliar. A Figura 4.1 apresenta uma foto tirada da vista superior da placa do módulo de baixa corrente, mostrando a disposição dos seus componentes eletrônicos, e a Figura 4.2 mostra uma foto tirada de sua parte inferior, destacando suas trilhas e soldas.

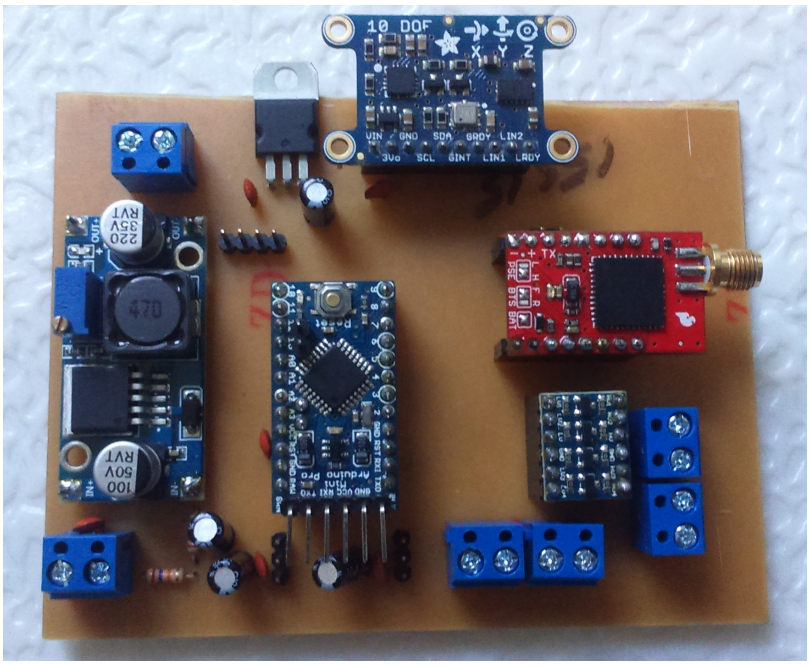


Figura 4.1: Vista superior da placa do módulo de baixa corrente.

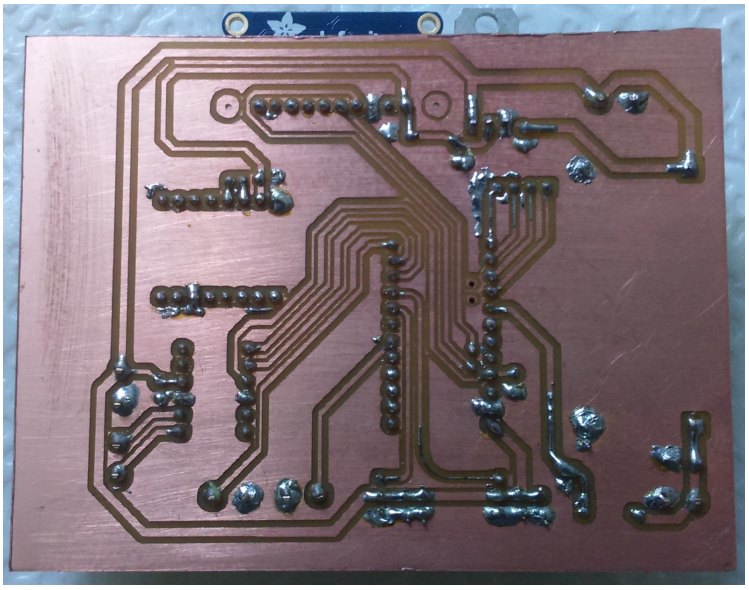


Figura 4.2: Vista inferior da placa do módulo de baixa corrente.

Em ambas as fotos, é possível observar que os elementos foram organizados com o objetivo de retirar qualquer fio interligando as trilhas. Para realizar a conexão com os componentes externos à placa, foram utilizados conectores com parafusos de duas entradas (na cor azul, nas bordas da placa), que garantem que os fios permaneçam sempre fixos e bem presos. Como relatado na seção 3.2.4, os diferentes comprimentos das trilhas e a utilização do plano de terra podem ser visualizados na vista inferior da placa.

A Figura 4.3 mostra, na parte superior, os suportes utilizados para prender os LEDs na estrutura do robô, e na parte inferior, a placa do circuito auxiliar. Nesta, que realiza a distribuição da energia para os módulos de baixa e alta corrente, cabe destacar a chave de liga/desliga, os conectores parafusados, e os conectores amarelos do tipo XT60, que são usados para conexões de alta amperagem com a bateria Lipo e com os ESCs.

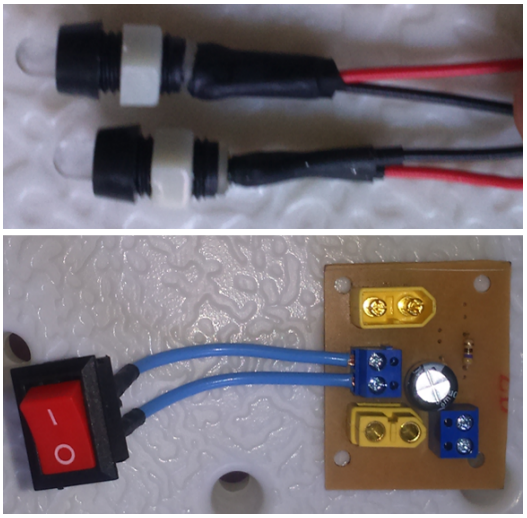


Figura 4.3: Suporte para os LEDS e placa do circuito auxiliar.

4.2 Projeto Mecânico

As Figuras 4.4 e 4.5 apresentam diferentes vistas do projeto mecânico finalizado, com todos os seus componentes encaixados. A primeira, mostra uma foto da vista em perspectiva, enquanto a segunda, uma foto da vista traseira.

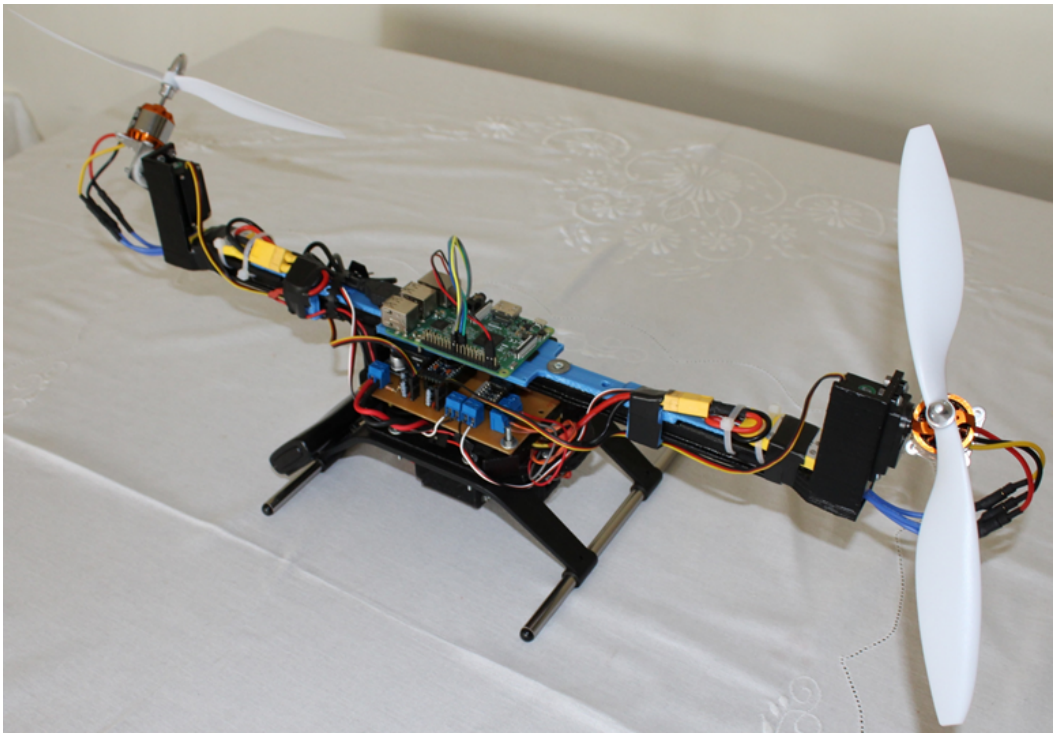


Figura 4.4: Vista em perspectiva do veículo aéreo.

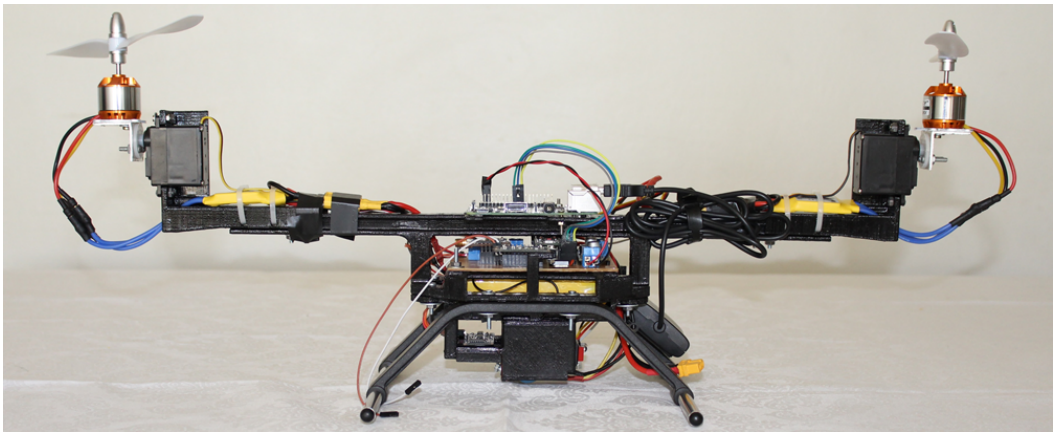


Figura 4.5: Vista frontal do veículo aéreo.

Assim como mostrado na seção 3.2.6, o novo protótipo de veículo aéreo construído procurou levar em consideração a organização e o posicionamento de cada elemento. A Figura 4.6 apresenta e enumera alguns detalhes que merecem destaque, são eles:

1. Fixação da unidade de medição inercial, garantindo que permaneça firme durante o voo;

2. Suporte para o sensor ultrassônico e para o circuito auxiliar localizado na parte inferior;

3. Acoplamento para a interface sem fio do *joystick*;

4. Encaixe para o botão de liga/desliga e para os LEDs de alto brilho.

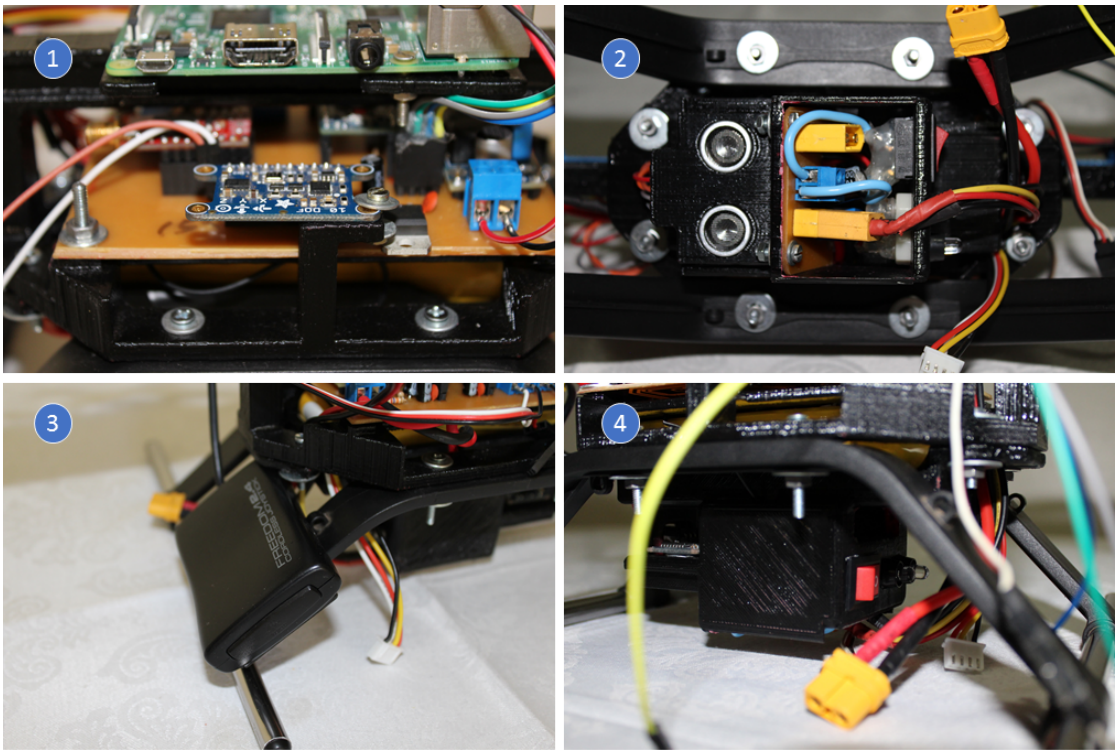


Figura 4.6: Detalhes estruturais do veículo aéreo.

Outro ponto a se destacar é a fixação dos ESCs e dos servomotores. Com um novo suporte, se obteve uma melhor fixação dos ESCs e uma canalização dos fios que os conectam com os rotores. Dessa forma, garantiu-se que os fios não atrapalhassem mais o funcionamento do servomecanismo e nem ficassem raspando no suporte dos rotores, o que poderia causar uma eventual ruptura. A Figura 4.7 destaca a ponta de uma das asas, mostrando, justamente, o novo suporte e a disposição obtida dos componentes e seus fios.

Utilizando uma balança de precisão, o peso da estrutura completa foi medido. Como mostra a Figura 4.8, o peso da estrutura com todos os seus elementos foi de 1112g. Portanto, para um empuxo máximo dos dois motores *brushless* de 1780g (890g cada), temos uma relação peso/empuxo de aproximadamente 62,47%; menor que a do protótipo anterior que era de 922g/1420g, ou seja, de aproximadamente 64,92%. Vale lembrar que o primeiro protótipo utilizava os rotores antigos, não possuía ainda um processador principal e nem uma placa de distribuição de energia. Como pretendido, o peso do veículo aéreo ficou abaixo de 80% do empuxo total fornecido pelos motores.

No Anexo II é possível visualizar o desenho técnico com todas as dimensões importantes do veículo aéreo.

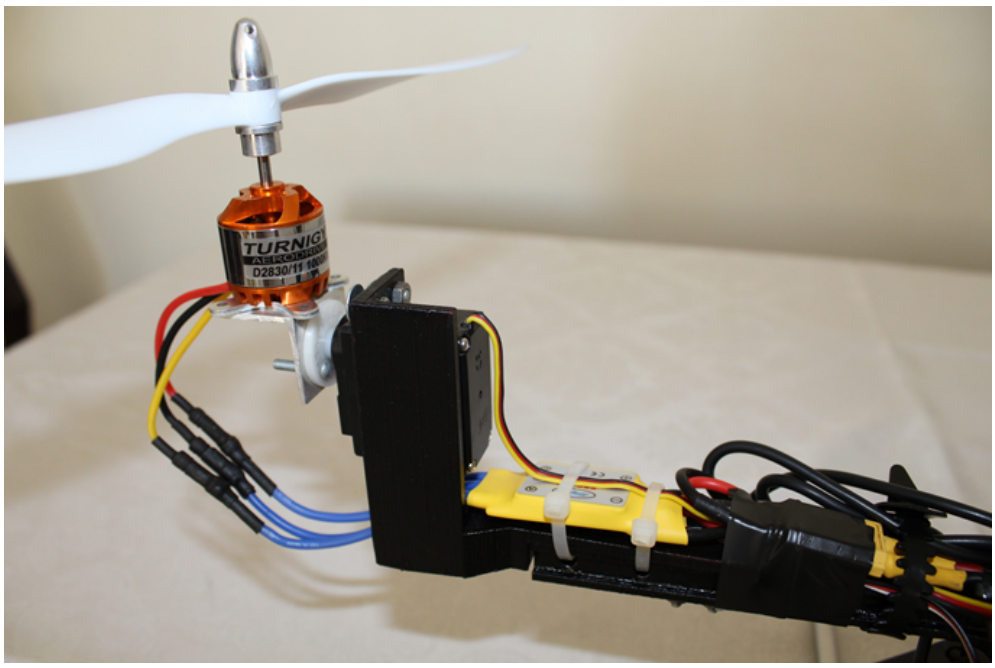


Figura 4.7: Detalhe estrutural do servomecanismo do veículo aéreo.

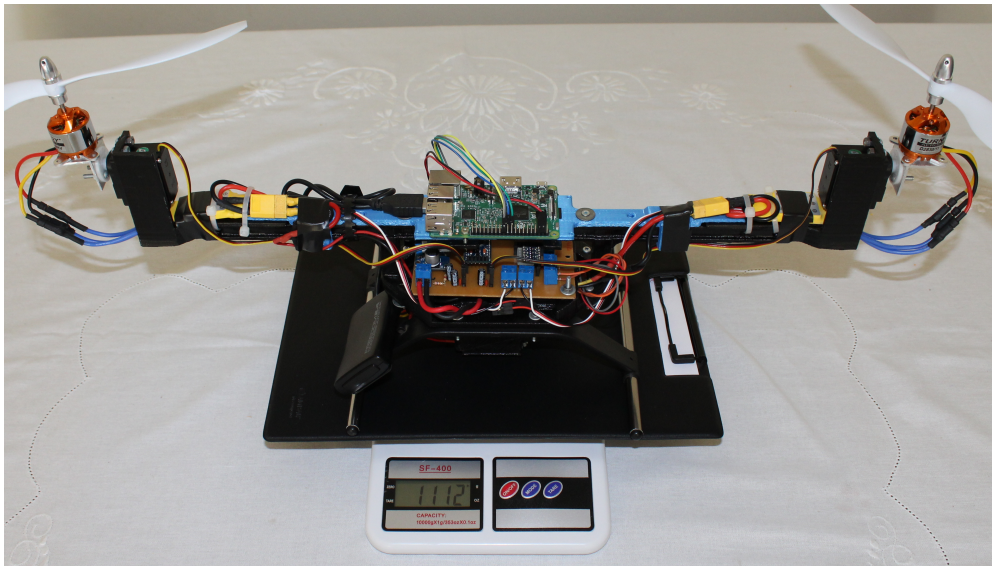


Figura 4.8: Peso medido do veículo aéreo.

4.3 Projeto do Controle

4.3.1 Estatísticas do *Software*

Como explicado na seção 3.3.1, a rotina de controle é executada na forma de um laço sem fim. A cada iteração, são armazenados os valores das variáveis escolhidas através do *Datalogger*. O gráfico apresentado na Figura 4.9 mostra a variação do intervalo entre as amostras, ao longo da execução do programa, em uma tentativa de voo realizada em ambiente externo. Cada amostra corresponde ao cálculo da diferença do tempo entre o valor atual, no instante t, e o valor imediatamente anterior,

no instante (t-1).

O período projetado para a rotina de controle foi de T = 0,020s, acima disso não seria adequado para o sistema e poderia comprometer o voo do veículo aéreo. A média aritmética do período de amostragem é de 16,61ms e segue destacada em vermelho no gráfico. Um pico de 194,4ms foi registrado, mas não foi descoberto o motivo durante a pesquisa. Acredita-se que a causa possa ser algum ruído ou algum atraso na comunicação SPI, visto que para que haja a troca de informações é necessário o sincronismo. Sem ele, o Mestre permanece tentando se comunicar até conseguir. A Tabela 4.1 mostra os dados estatísticos referentes ao gráfico da Figura 4.9.

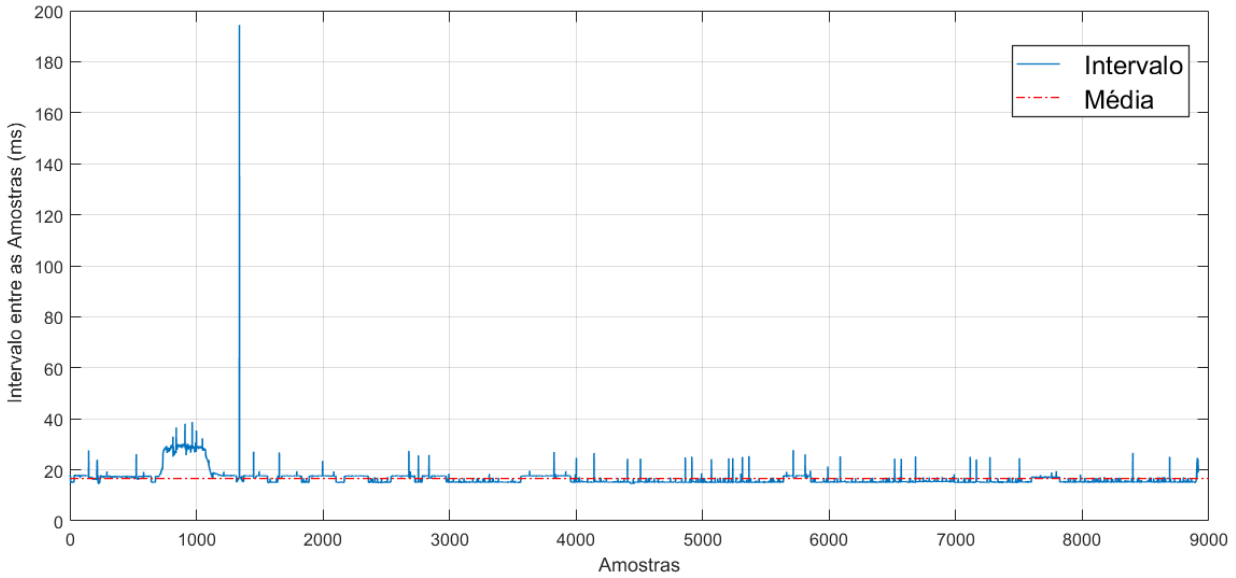


Figura 4.9: Período de amostragem.

Tabela 4.1: Estatística do período de amostragem

Tempo total de execução (s)	Intervalo entre as Amostras					
	Média (ms)	Mínimo (ms)	Máximo (ms)	Moda (ms)	Mediana (ms)	Desvio Padrão (ms)
148,1	16,61	14,42	194,4	15,11	15,35	3,36

4.3.2 Calibração dos Sensores

Realizando a leitura dos valores de velocidade angular fornecidos pelo girômetro com o veículo aéreo parado, duas coisas foram observadas: havia um deslocamento ou *offset* nas medidas, e também um ruído que gerava uma variação na leitura. O gráfico na Figura 4.10 mostra as medições feitas em cada um dos eixos com a aeronave parada e com seus motores sendo acionados em baixa potência (29,89%) e depois desligados.

Por meio de uma análise estatística, apresentada na Tabela 4.2, foi feita uma compensação do

offset em cada eixo, via *software*, a partir da média dos valores medidos. Além disso, se utiliza um filtro passa-faixa para retirar a parte ruidosa, colocando em zero os valores dentro da faixa indicada pelo desvio padrão.

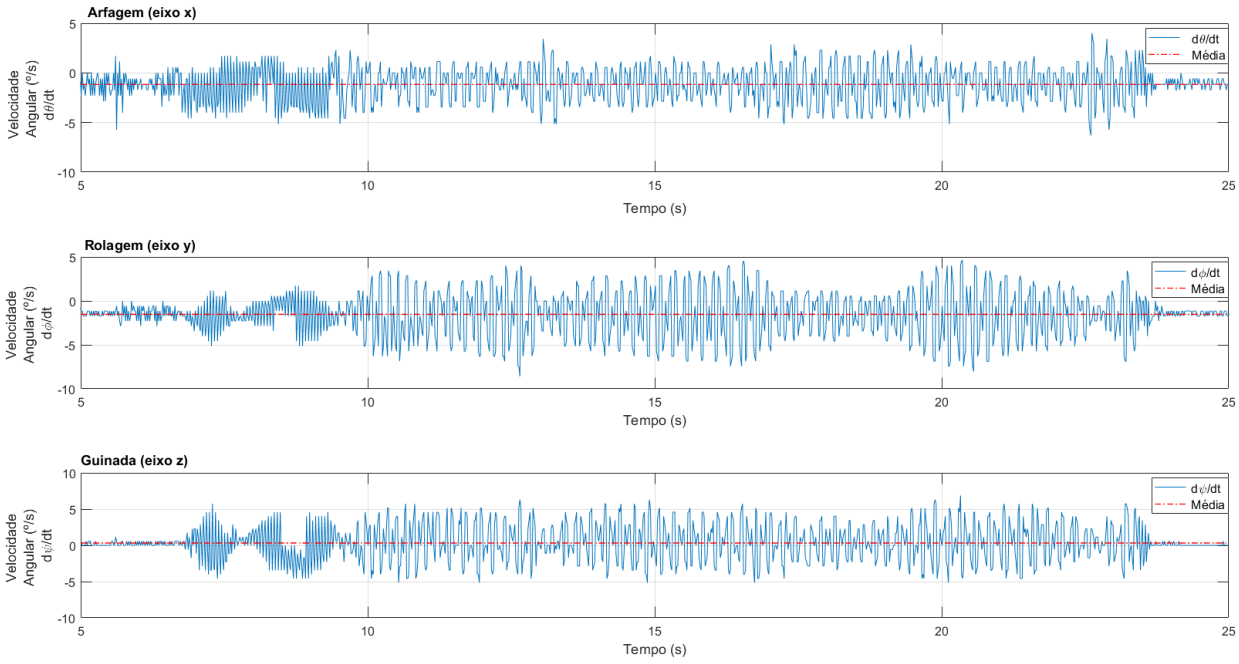


Figura 4.10: Velocidades angulares nos três eixos de movimento com o birrotor parado.

Tabela 4.2: Dados estatísticos das velocidades angulares nos três eixos de movimento com o birrotor parado

Eixos de Movimento	Velocidade Angular					
	Média (°/s)	Mínimo (°/s)	Máximo (°/s)	Moda (°/s)	Mediana (°/s)	Desvio Padrão (°/s)
Arfagem	-1,16	-6,303	4,011	-1,146	-1,146	1,645
Rolagem	-1,522	-8,594	4,584	-1,146	-1,146	2,353
Guinada	0,3206	-5,157	6,875	0	0	2,326

A Figura 4.11 mostra o resultado das implementações em um teste realizado em ambiente externo com o veículo no solo. O gráfico representa as velocidades angulares medidas ao longo do tempo, considerando o *offset* e também a filtragem de pequenos ruídos. Da mesma forma, foi realizada uma análise estatística, a qual está exposta na Tabela 4.3. Nota-se neste novo gráfico que houve uma melhora em relação ao *offset* das medidas, visto que os valores estão próximos de zero, e que além disso, ruídos de baixa frequência também foram filtrados.

Com o sensor ultrassônico foi realizado o mesmo procedimento com a aeronave no solo. As Figuras 4.12 e 4.13 mostram dois gráficos, o inicial e o obtido após a compensação do *offset*,

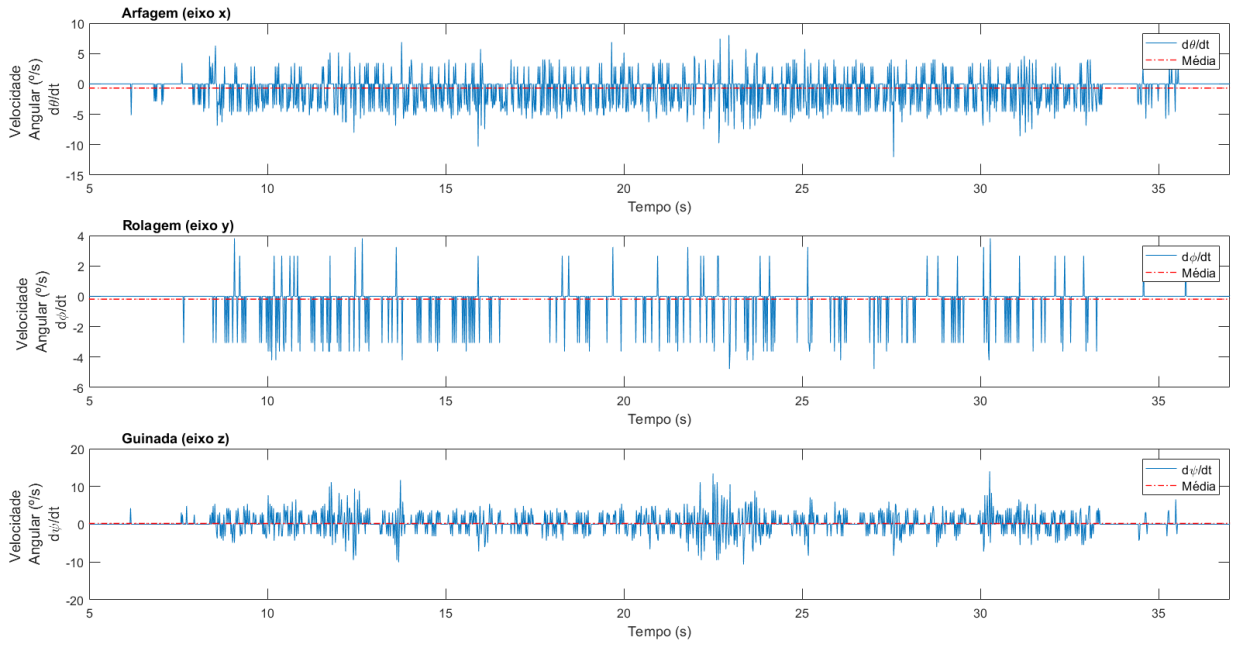


Figura 4.11: Velocidades angulares nos três eixos de movimento com o birrotor parado após alterações.

Tabela 4.3: Dados estatísticos das velocidades angulares nos três eixos de movimento com o birrotor parado após alterações

Eixos de Movimento	Velocidade Angular					
	Média (°/s)	Mínimo (°/s)	Máximo (°/s)	Moda (°/s)	Mediana (°/s)	Desvio Padrão (°/s)
Arfagem	-0,6884	-12,03	8,025	0	0	2,229
Rolagem	-0,1895	-4,783	3,812	0	0	0,9338
Guinada	0,2003	-10,63	14	0	0	2,232

respectivamente. O deslocamento no sonar é explicado pela distância do suporte do sensor até a superfície, e não apenas decorrente de algum ruído ou erro de leitura, como o caso do girômetro. Portanto, a alteração é implementada para considerar o veículo aéreo na superfície como possuindo uma altura zero. A altura real do sensor é de 2cm da superfície e em um primeiro momento a média do valor medido foi de 3,367cm. Aplicando-se esse valor como *offset*, e realizando novos testes, foi obtida uma nova média de 0,1121cm. A análise estatística de antes e depois da correção são apresentadas na Tabela 4.4. É importante ressaltar que os valores abaixo de zero foram suprimidos por meio do *software*, pois são fisicamente impossíveis.

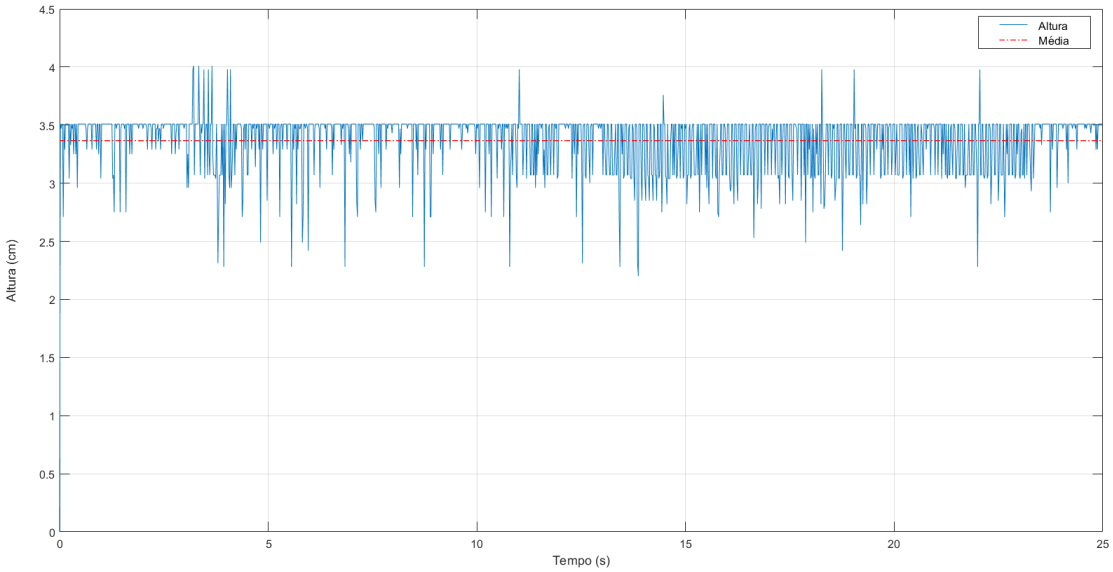


Figura 4.12: Medidas de altura realizadas pelo sonar.

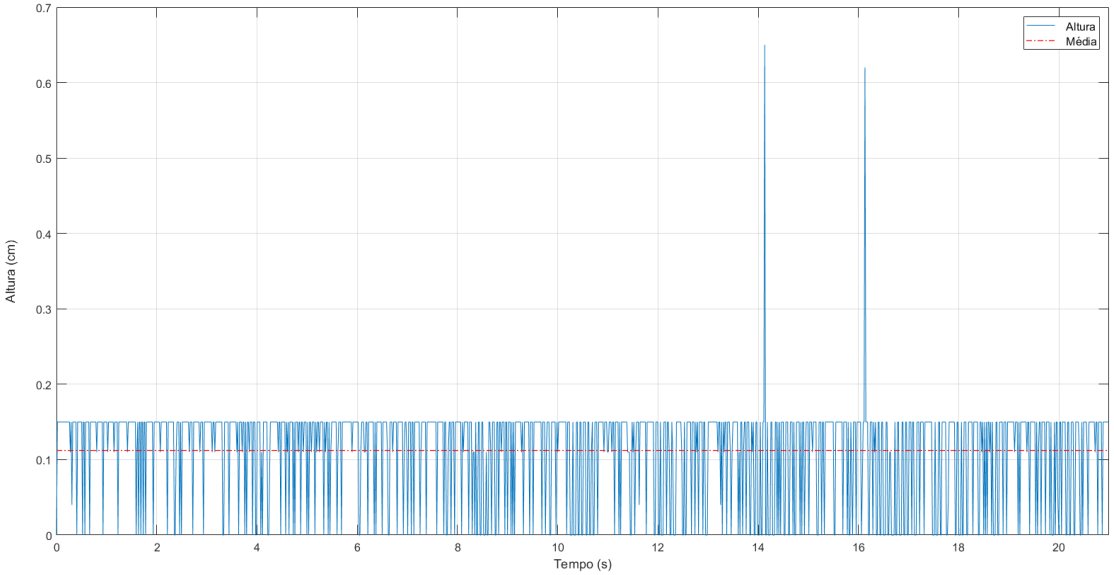


Figura 4.13: Medidas de altura realizadas pelo sonar após alterações.

Tabela 4.4: Dados estatísticos das medidas de altura realizadas pelo sonar antes e depois da correção

Situação	Tempo total de execução (s)	Altura					
		Média (cm)	Mínimo (cm)	Máximo (cm)	Moda (cm)	Mediana (cm)	Desvio Padrão (cm)
Antes	25	3,367	0	4,01	3,51	3,51	0,2632
Depois	22	0,1121	0	0,65	0,15	0,15	0,0665

4.3.3 Leitura da Bateria

A leitura da bateria é feita para identificar quando a carga está em um nível crítico, indicando quando deve ser realizada a recarga. A medição da tensão na bateria é uma tarefa periódica realizada de 5 em 5 segundos, já que não há necessidade de realizá-la de maneira ininterrupta. A tensão máxima da bateria Lipo utilizada é de 12,6V, que corresponde a 4,2V por célula. A tensão estabelecida para acionar o alarme foi de 10,8V, o equivalente a uma tensão de 3,6V por célula. Em uma bateria Lipo, a taxa de descarga é alta no ínicio, se estabiliza na região central e aumenta de novo em um nível crítico de tensão. O valor de 10,8V foi escolhido por ser antes dessa queda repentina. A Figura 4.14 mostra o gráfico obtido em uma tentativa de voo realizado em ambiente externo, retratando a variação da tensão da bateria Lipo em volts ao longo do tempo de execução em segundos. Embora a descarga de uma bateria desse tipo não seja linear, a região central de tensão pode ser linearizada para efeito de aproximação. Assim foi feita uma regressão linear no intervalo avaliado, obtendo a seguinte equação:

$$V_{Bateria} = -0,00414t + 11,6927. \tag{4.1}$$

Avaliando a equação obtida é possível estimar nesse caso, mais ou menos, quanto tempo ainda teria de bateria até chegar ao nível crítico estabelecido de 10,8, o que daria aproximadamente 215 segundos. A medição na tensão da bateria apresenta bastante ruído, por isso foi determinado através do *software* que o alarme só seja acionado quando o limite de 10,8V for ultrapassado 5 vezes seguidas.

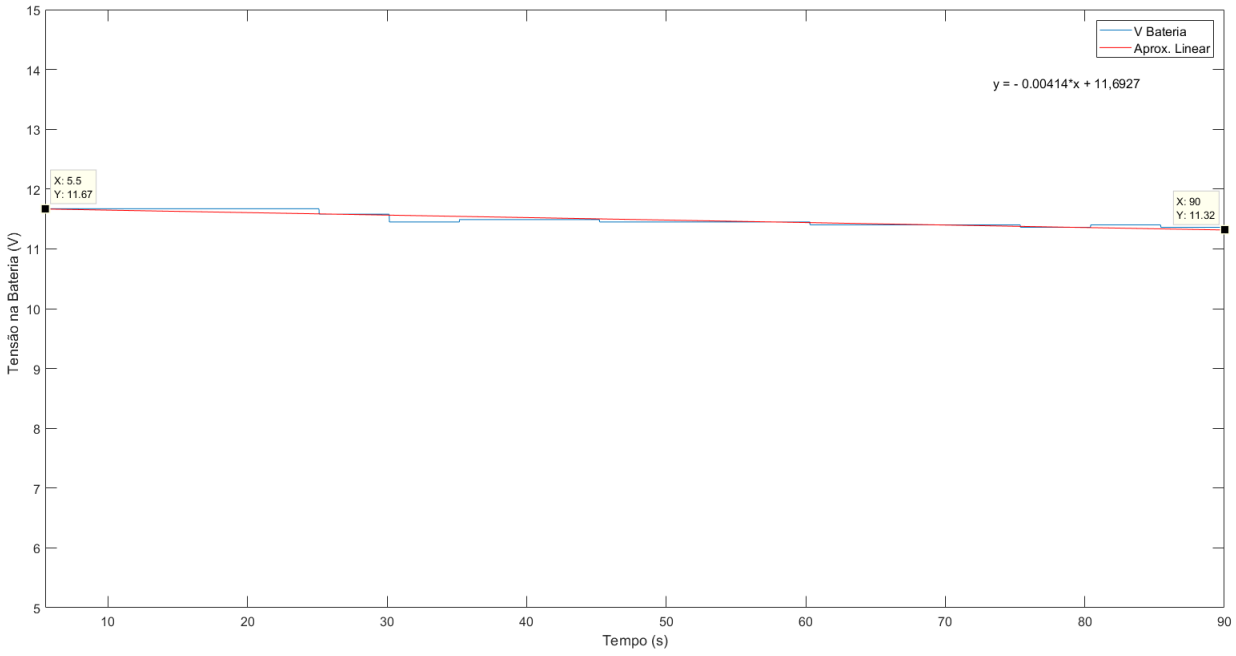


Figura 4.14: Medição da tensão na bateria Lipo.

4.3.4 Modo *Joystick*

Na ação de controle Proporcional (P) implementada no modo *Joystick*, o ajuste do ganho K_P foi realizado através de tentativa e erro até se perceber uma melhora na interface com o usuário. A sensibilidade foi diminuída na interpretação dos comandos enviados tentando sempre proporcionar uma sensação melhor de controle, evitando alterações bruscas de orientação, assim como explicado na página 55. O ganho K_P escolhido foi de 0,2 e assim, baseado nas equações (3.3), (3.4), (3.5) e (3.6), as leis de controle para esse modo ficaram definidas como:

$$Sd_t = Sd_{t-1} + 0,2(Sd_{des} - Sd_{t-1}), \tag{4.2}$$

$$Se_t = Se_{t-1} + 0,2(Se_{des} - Se_{t-1}), \tag{4.3}$$

$$Pd_t = Pd_{t-1} + 0,2(Pd_{des} - Pd_{t-1}), \tag{4.4}$$

$$Pe_t = Pe_{t-1} + 0,2(Pe_{des} - Pe_{t-1}). \tag{4.5}$$

As próximas três figuras mostram os gráficos que representam essas equações de controle em cada eixo. Note que cada gráfico é duplo: no eixo vertical esquerdo estão retratadas as variações do comando de entrada do *joystick* ao longo do tempo; enquanto no eixo vertical direito estão retratadas as variações da saída medidas do sistema ao longo do tempo. Nas Figuras 4.15 e 4.16, o eixo vertical direito corresponde à posição angular dos servos, e na Figura 4.17, ele retrata a potência dos rotores.

A Figura 4.15 apresenta o gráfico de um movimento de guinada, gerado pela variação do comando do *joystick* no eixo Z. Nesse tipo de movimento, os rotores são virados em sentidos opostos; entretanto, note no gráfico que a posição angular dos servos direito e esquerdo andam sempre juntas, isso ocorre porque suas referências estão invertidas e a variação angular β terá sempre o mesmo sinal.

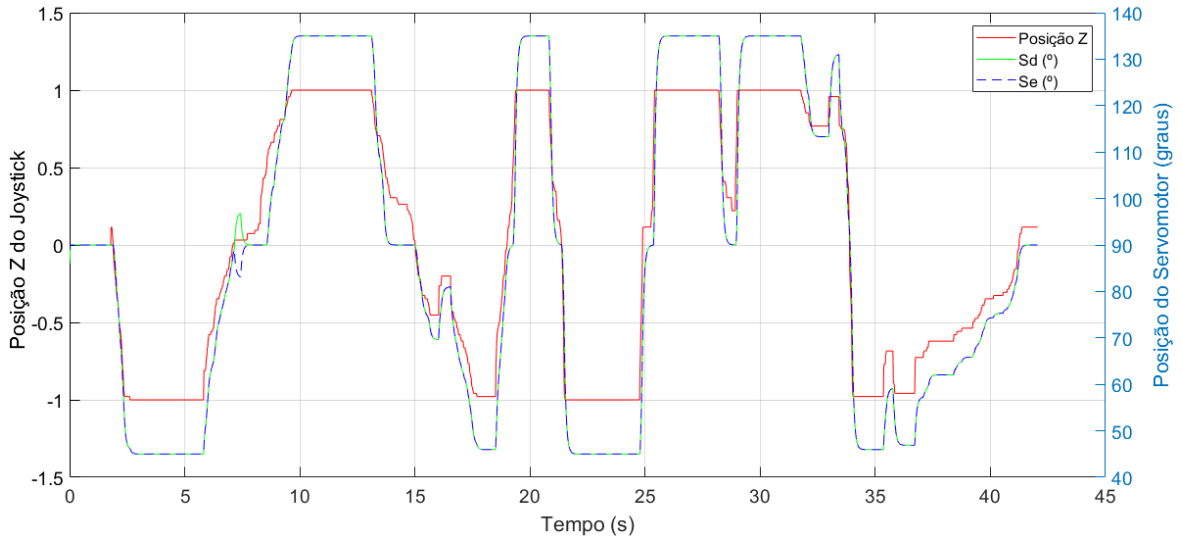


Figura 4.15: Controle P de guinada no Modo *Joystick*.

A Figura 4.16 mostra o movimento de arfagem, com a variação no eixo Y do *Joystick*. Nesse movimento, os rotores são movimentados juntos para frente ou para trás, o que é facilmente observado no gráfico por causa da simetria entre as posições angulares do servo direito e do servo esquerdo. Devido às referências inversas, durante a arfagem a variação angular β do servo direito possuirá um sinal oposto ao do servo esquerdo.

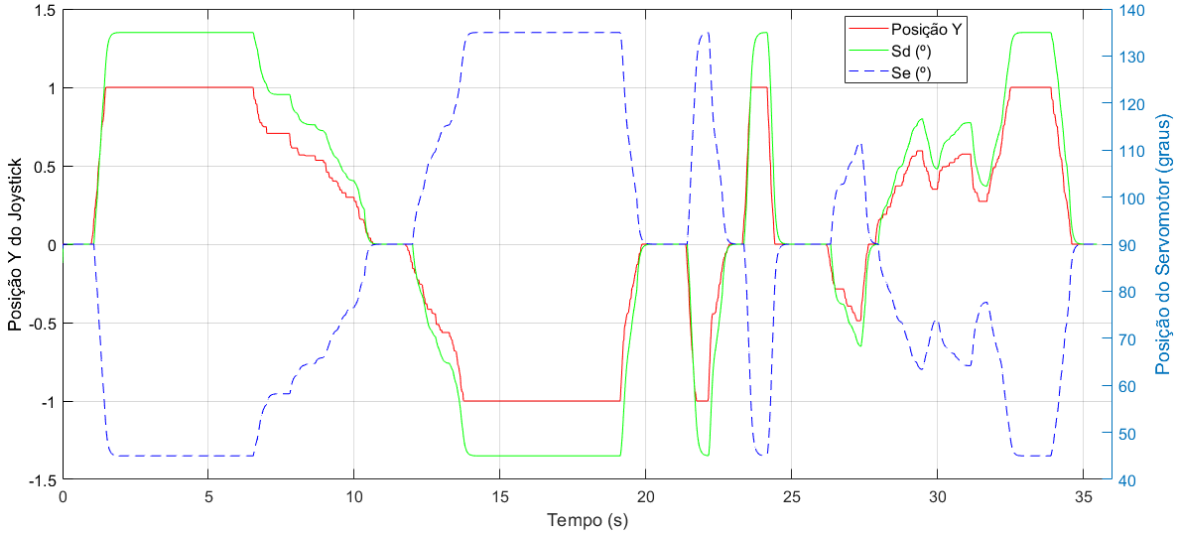


Figura 4.16: Controle P de arfagem no Modo *Joystick*.

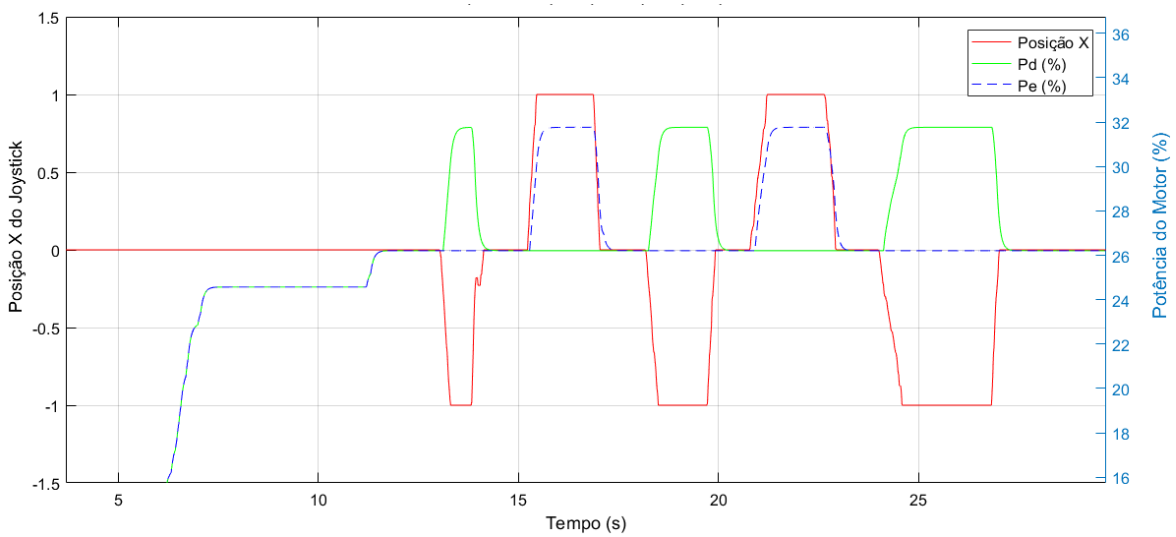


Figura 4.17: Controle P de rolagem no Modo *Joystick*.

Por último, a Figura 4.17 retrata o movimento de rolagem, provocado pela variação da entrada do *Joystick* no eixo X. Nesse movimento, há uma alteração nas velocidades dos rotores para controlar o giro. Como pode ser visualizado no gráfico, o comando para a esquerda no *Joystick* (valores negativos) aumenta a potência do rotor direito, o que faz o veículo aéreo virar para a esquerda, e da mesma forma, o comando para a direita aumenta a potência do rotor esquerdo, o que faz com que ele vire para a direita.

4.3.5 Modo de Estabilização

No modo de Estabilização foi implementada uma lei de controle Proporcional-Derivativo (PD) para estabilizar o sistema, com o objetivo maior de realizar o controle de atitude e conseguir atingir a estabilização do veículo. Assim como mostrado em [29], em um veículo aéreo birrotor articulado, um simples controlador proporcional não é suficiente para levar o sistema à estabilidade, principalmente por causa da sua dinâmica rápida e subatuada, com apenas 4 atuadores para 6 graus de liberdade. Dessa forma, a ação de controle PD deve possuir ganhos capazes de estabilizar o veículo de forma rápida, mas sem gerar uma oscilação grande.

Por meio de uma análise direta utilizando o método de tentativa e erro, diferentes valores de ganhos para K_{P1}, K_{P2}, K_{D1}, K_{D2} e K_{D3}, definidos nas equações de (3.15) a (3.18), foram experimentados. Encontrar valores que apresentassem uma boa resposta em um ambiente controlado não foi uma grande dificuldade, mas sim quando o veículo era levado para testes de voo em campo aberto; situação em que os rotores estão em uma potência muito elevada e qualquer variação angular de β gera uma força de propulsão muito grande no sentido aplicado. Dito isso, percebeu-se que uma leve inclinação do rotor a uma velocidade de rotação elevada fazia com que o veículo se balançasse desgovernadamente.

A ação de controle derivativa atua de acordo com a variação do erro, produzindo uma correção antes que a magnitude do erro se torne muito grande. Pensando nisso e analisando as leituras do girômetro obtidas em testes que não deram certos, notou-se que com a potência elevada dos rotores, o valor da velocidade angular aumentava muito com uma simples inclinação. Dessa forma, o valor do ganho derivativo foi reduzido, diminuindo a sensibilidade do sistema em relação ao giro em cada eixo e aumentando a robustez do controle. A redução do ganho trouxe resultados positivos em tentativas de voo, apresentando uma maior estabilidade no sistema. Outro ponto a se destacar era a presença de valores muito altos de giro; logo, para se criar situações de teste foi incluído um saturador no programa que limitou o giro em cada um dos eixos até uma velocidade de 100^{o}/s.

Para atingir uma condição de voo é necessária uma potência em torno de 64%, valor esse medido em testes controlados com o birrotor preso. Com ele solto e com uma potência um pouco inferior, foi ativado o modo de controle automático e foram criadas situações que imitassem cada um dos movimentos em cada eixo. Dessa forma, os valores para os ganhos foram escolhidos e as leis de controle ficaram definidas na forma

$$Sd_t = Sd_{t-1} + 0,77(Sd_{des} - Sd_{t-1}) + 0,12(\dot{\psi}_{des} - \dot{\psi}_{t-1}) + 0,06(\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{4.6}$$

$$Se_t = Se_{t-1} + 0,77(Se_{des} - Se_{t-1}) + 0,12(\dot{\psi}_{des} - \dot{\psi}_{t-1}) - 0,06(\dot{\theta}_{des} - \dot{\theta}_{t-1}), \tag{4.7}$$

$$Pd_t = Pd_{t-1} + 0,77(Pd_{des} - Pd_{t-1}) + 0,025(\dot{\phi}_{des} - \dot{\phi}_{t-1}), \tag{4.8}$$

$$Pe_t = Pe_{t-1} + 0,77(Pe_{des} - Pe_{t-1}) - 0,025(\dot{\phi}_{des} - \dot{\phi}_{t-1}). \tag{4.9}$$

O controle de rolagem foi colocado como o menos sensível ao giro, visto que um giro muito alto aumentaria muito a potência de um dos motores e isso levaria a uma situação desastrosa.

Com esses valores definidos, foram obtidos os próximos três gráficos apresentados nas Figuras 4.18, 4.19 e 4.20, todos obtidos em uma condição de potência pré-voo. Todos os três gráficos são duplos, o eixo vertical da esquerda representa a velocidade angular medida pelo girômetro e o eixo vertical da direita apresenta a posição angular dos servomotores, ambos medidos ao longo do tempo de execução do programa. Os três gráficos abrangem o mesmo intervalo de tempo. A Figura 4.18 retrata o controle de posição do servomotor direito, a Figura 4.19 retrata o mesmo para o servomotor esquerdo, e a Figura 4.20 mostra os dois juntos ampliados entre os instantes de 83 a 87 segundos.

Nos três gráficos, a cor vermelha corresponde à velocidade angular de guinada, enquanto a cor azul corresponde à velocidade angular de arfagem. No primeiro e no segundo é possível observar a compensação dos servos na tentativa de anular o movimento de guinada, principalmente entre os instantes de tempo 77s e 83s. Já na ampliação apresentada no gráfico da Figura 4.20 se observa a resposta dos dois servos a ambos os movimentos de guinada e arfagem, nos quais as ações de controle se somam para compensarem cada um dos movimentos.

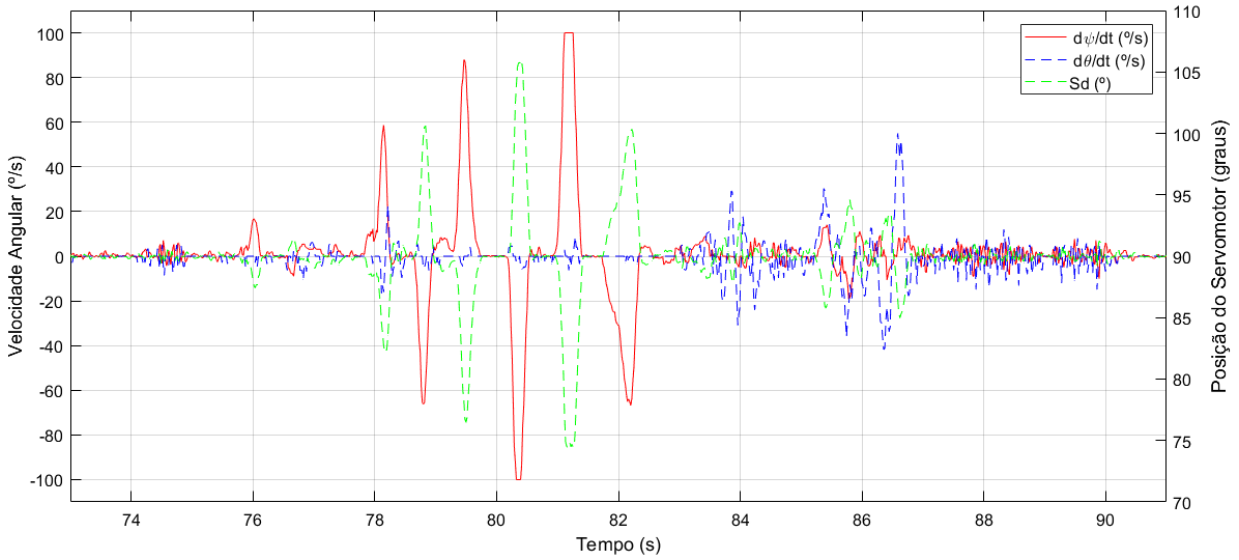


Figura 4.18: Controle PD aplicado no servo direito.

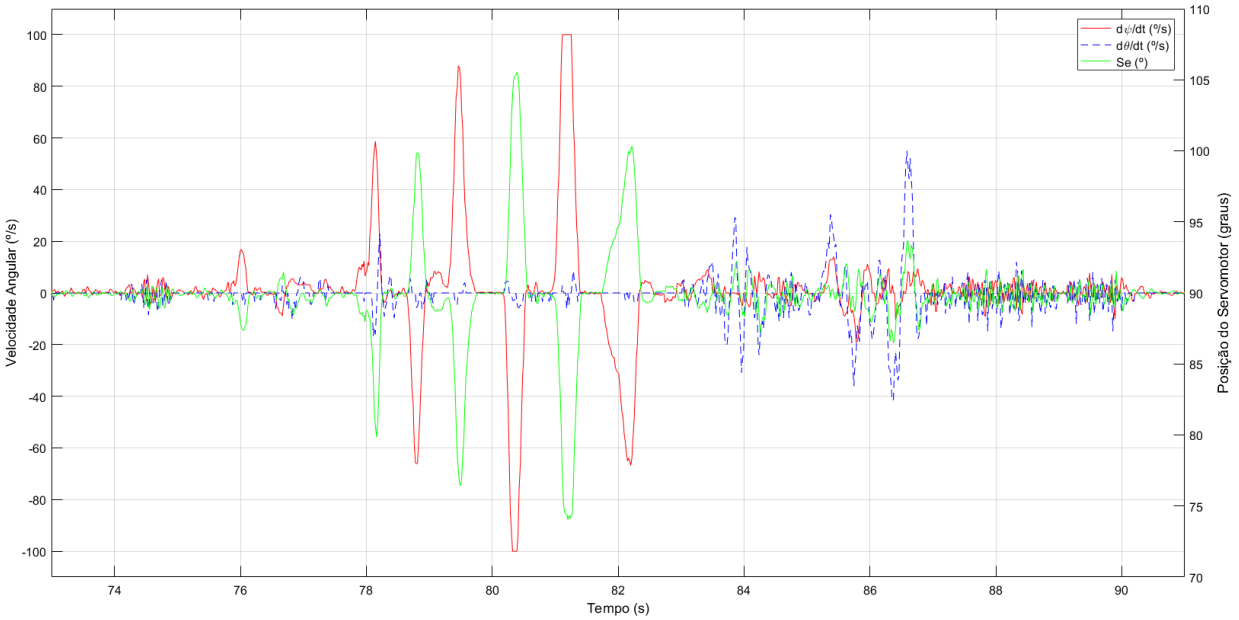


Figura 4.19: Controle PD aplicado no servo esquerdo.

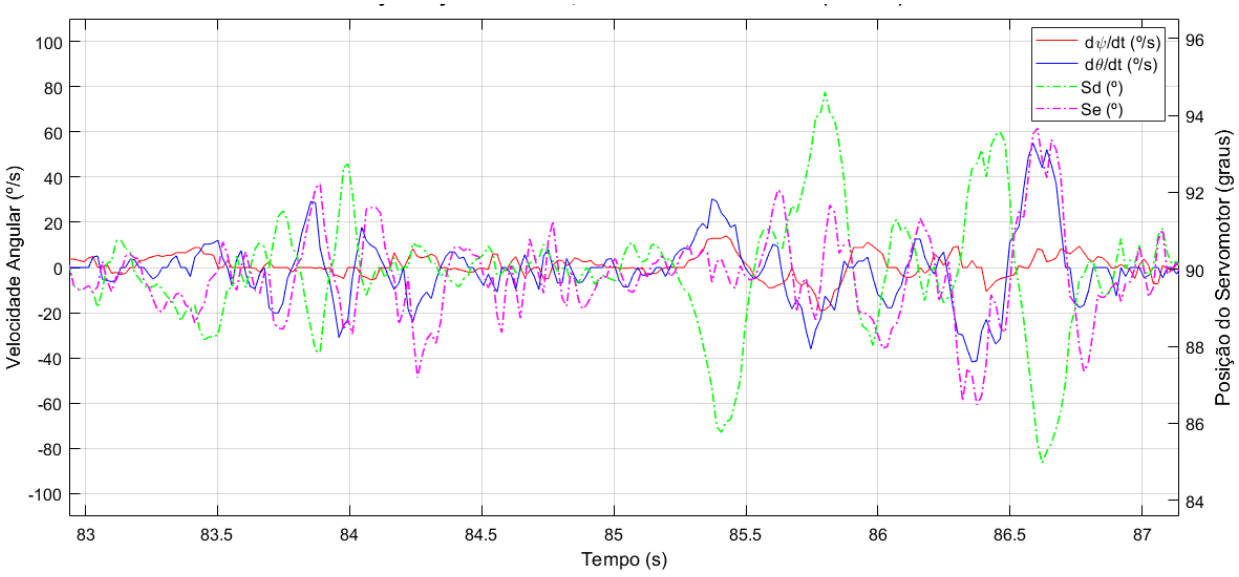


Figura 4.20: Controle PD aplicado nos servos esquerdo e direito.

O gráfico apresentado na Figura 4.21 mostra o controle PD aplicado no movimento de rolagem, controlando a potência nos rotores direito e esquerdo. No eixo vertical esquerdo é retratada a velocidade angular no eixo y, enquanto no eixo vertical direito é mostrada a potência dos rotores em porcentagem. É possível observar o sistema compensando o giro alterando a potência em cada rotor. No gráfico da Figura 4.21 a cor vermelha indica a velocidade angular no eixo y, a cor verde indica a potência do rotor direito e a cor azul a potência do rotor esquerdo.

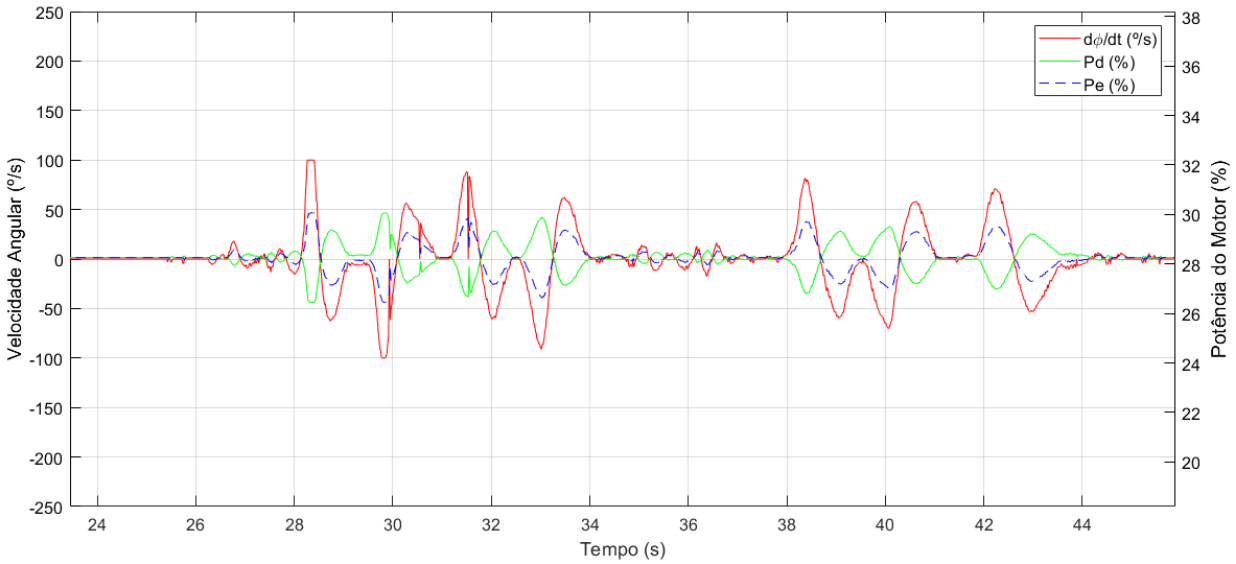


Figura 4.21: Controle PD aplicado nos rotores esquerdo e direito.

4.3.6 Tentativa de Voo

Diante dos resultados expostos, o veículo aéreo birrotor foi levado a um ambiente aberto para testes de voo. Uma tentativa utilizando o modo de Estabilização foi bem sucedida; a aeronave conseguiu alçar voo com sucesso, mas devido à inexperiência do piloto, não obteve uma boa aterrisagem. A Figura 4.22 mostra uma foto tirada no teste. Um vídeo do voo foi feito e os dados obtidos são mostrados nos próximos dois gráficos. O primeiro, na Figura 4.23, apresenta o controle PD aplicado no servomotor direito (verde) durante o período de voo, tentando compensar as velocidades angulares nos movimentos de guinada (vermelho) e arfagem (azul). O segundo, na Figura 4.24, apresenta o controle PD aplicado nos rotores esquerdo (azul) e direito (verde), tentando compensar a velocidade angular causada pelo movimento de rolagem (vermelho). A resposta era muito boa visualmente, porém uma tentativa de pouso feita pelo piloto a partir do *Joystick*, fez com que a potência dos rotores diminuísse muito. Infelizmente, a redução foi demais e isso causou uma queda no veículo. É possível ver a redução da potência no gráfico a partir do instante de 65s. Devido ao elevado número de quedas já sofridas anteriormente, algumas pequenas fraturas na estrutura foram identificadas, mas nada que comprometesse totalmente o veículo. Recomenda-se para futuros testes avaliar a estrutura, e se possível fabricá-la novamente.

Figura 4.22: Foto tirada durante voo do VANT.

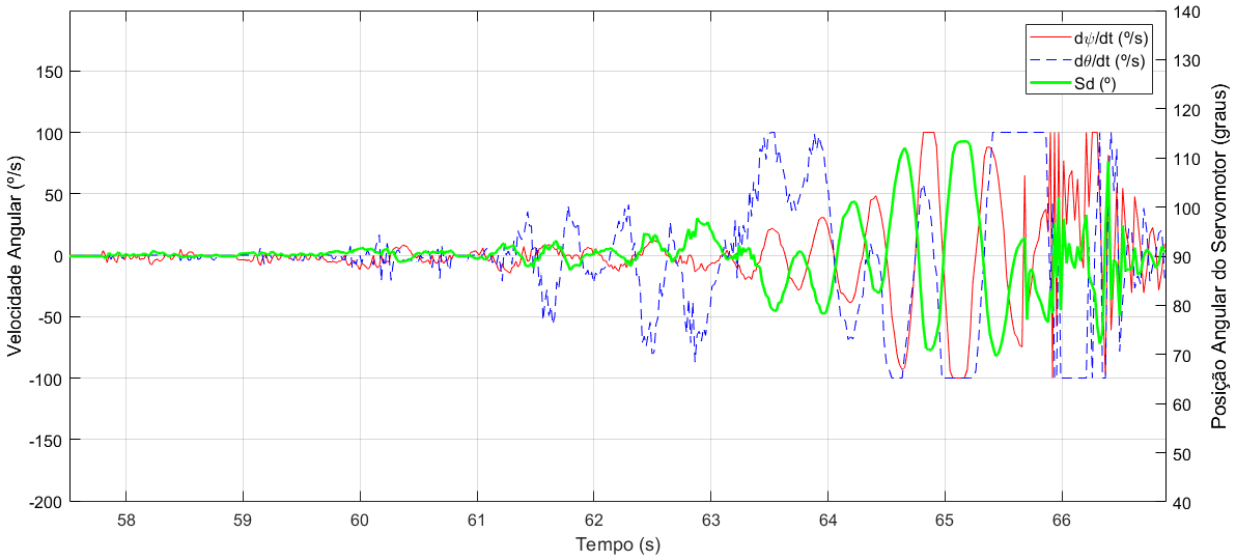


Figura 4.23: Controle PD aplicado no servo direito durante um voo.

4.4 Monitoramento Remoto

Através da plataforma de *software* VNC *Viewer*[1] da empresa Real VNC, foi possível realizar um acompanhamento remoto do funcionamento do programa implementado por meio de um computador pessoal, que serviu como estação de controle. A plataforma permite a conexão via

[1]VNC *Viewer*, disponível em: <https://www.realvnc.com/en/connect/download/viewer/>.

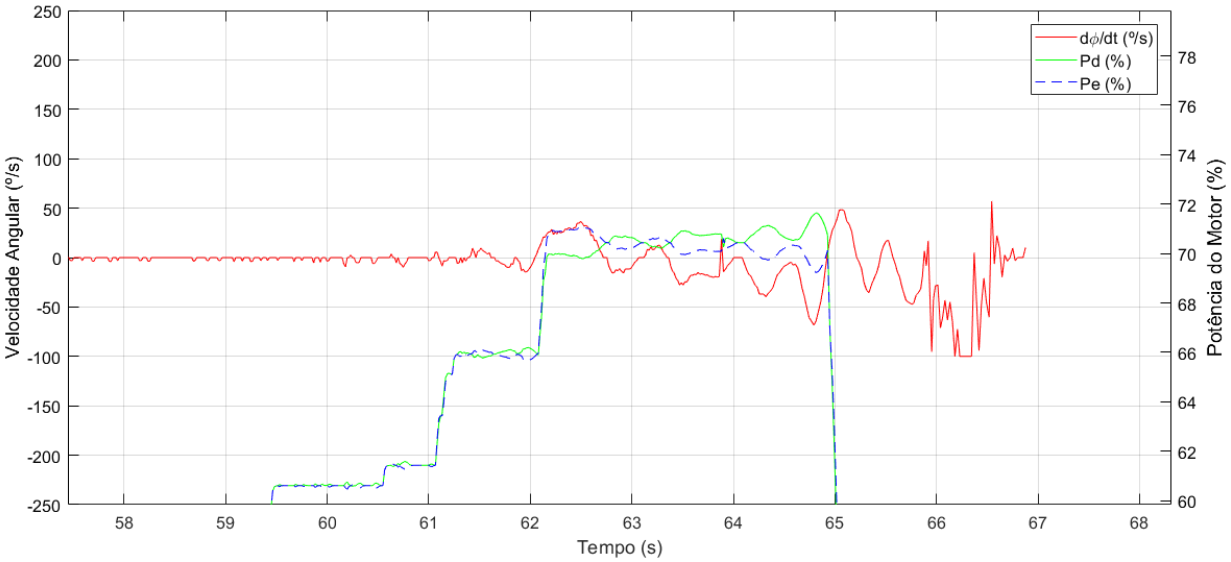


Figura 4.24: Controle PD aplicado nos rotores esquerdo e direito durante um voo.

SSH utilizando cabo de rede e também uma conexão sem fio, caso o veículo aéreo e o computador pessoal estejam conectados na mesma rede.

Capítulo 5

Conclusões

Um veículo aéreo *tilt rotor* é uma aeronave que consegue realizar voos estáveis de baixa velocidade, decolar e pousar na vertical como helicópteros, e atingir velocidades maiores de forma semelhante a um avião. Por isso é uma aeronave híbrida e possui bastante futuro, visto que apresenta uma adaptabilidade para diferentes situações. Com seis graus de liberdade no total, é considerado um sistema subatuado por possuir apenas 4 atuadores, sendo dois servomotores e dois motores sem escovas. Diante disso e de sua dinâmca rápida e complexa, se torna um grande desafio a ser trabalhado.

Este trabalho apresentou uma nova etapa do desenvolvimento do veículo aéreo não tripulado do tipo *tilt rotor* com dois rotores articulados do Laboratório de Robótica Aérea da Universidade de Brasília. Todo o seu projeto mecânico, eletrônico, de *software* e de controle foram trabalhados, resultando em um novo protótipo com um sistema embarcado baseado em Linux, e com uma estrutura mais próxima de um produto fechado. Com todos os módulos de sensoreamento funcionando e com uma fonte de energia capaz de dar autonomia de voo, foram implementados dois modos de controle. O primeiro, chamado de Modo *Joystick*, utiliza um controlador Proporcional que atua no controle do veículo através da utilização de um *joystick* pelo usuário. O segundo, chamado de Modo de Estabilização, utiliza um controlador Proporcional-Derivativo para realizar o controle da dinâmica de orientação do veículo nos três eixos de movimento: na guinada, na arfagem e na rolagem.

Inúmeros testes foram realizados em ambiente controlado e em ambiente externo. Com esses testes, os valores dos ganhos dos controladores foram escolhidos e um voo conseguiu ser realizado com sucesso, mostrando que a abordagem utilizada está no caminho certo para controlar a atitude e a orientação da aeronave.

Entende-se que, ao final deste projeto seus objetivos estabelecidos foram cumpridos, levando o veículo aéreo para um novo estágio de desenvolvimento, no qual voar não é mais um pensamento e sim uma realidade.

Espera-se que o desenvolvimento da aeronave seja continuado e, para isso, alguns pontos são destacados para trabalhos futuros:

- Uma avaliação estrutural é necessária devido às quedas durante os testes. A fabricação das peças com 100% de preenchimento seria o ideal para garantir a resistência da estrutura;
- A modelagem 3D e a fabricação de uma carcaça para proteger os componentes internos do veículo;
- O aprimoramento do *software* é uma tarefa sempre necessária;
- A implementação da *thread* de leitura do GPS;
- O aprimoramento do controle de atitude, e o desenvolvimento do controle de altitude e do controle de posição, permitindo implementar um controle da dinâmica translacional do sistema;
- Realizar a união entre o Modo *Joystick* e o Modo de Estabilização, para que melhore a interface de controle para o piloto através do *joystick*.
- Utilizando a bateria Lipo, as medições da tensão apresentam bastante ruído. Pesquisar, identificar e corrigir esse problema;
- Pesquisar e aplicar métodos de filtragem na leitura do girômetro e dos sensores, que possuem ruídos, principalmente devido à rotação dos rotores.
- Desenvolver uma interface de monitoramento remoto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MAISEL, D. J. G. M. D.; DUGAN, D. C. *The History of the XV-15 Tilt Rotor Research Aircraft. From Concept to Flight. The Nasa History Series*. National Aeronautics and Space Administration: NASA, 2000. 197 p.

[2] JUNG, D. H.; CARVALHO, R. de L. *SLAM Visual Monocular baseado em Filtro de Kalman Estendido para Robôs Aéreos*. Trabalho de Graduação em Engenharia de Controle e Automação. Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF: UnB, 2016.

[3] CATSOULIS, J. *Designing Embedded Hardware: Create New Computers and Devices*. Second edition. [S.l.]: O'Reilly Media, Inc., 2005.

[4] BAI, Y. *Practical Microcontroller Engineering with ARM Technology*. [S.l.]: John Wiley & Sons, 2015.

[5] VALE, D. F.; COSTA, M. Ângelo A. da. *Implementação do Hardware e Software de estabilização de um Robô Aéreo com dois Rotores Articulados*. Trabalho de Graduação em Engenharia de Controle e Automação. Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF: UnB, 2016.

[6] BEARD, R. W.; MCLAIN, T. W. *Small unmanned aircraft: Theory and practice*. Princeton University: Princeton university press, 2012.

[7] CHOWDHURY, A. K. A. B.; RAINA, G. A generalized control method for a tilt-rotor uav stabilization. *Proceedings of the 2012 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, p. 309–314, maio 2012.

[8] WU, W.-Z. L. H.-S.; WANG, Y.-L. Research on the structure design and flight performance os a small tilt rotor unmanned aircraft. *Chinese Automation Congress (CAC)*, p. 1979–1985, nov. 2015.

[9] KENDOUL, F.; FANTONI, I.; LOZANO, R. Modeling and control of a small autonomous aircraft having two tilting rotors. *IEEE Transactions on Robotics*, IEEE, v. 22, n. 6, p. 1297–1302, 2006.

[10] LI, Q.; YAO, C. *Real-time concepts for embedded systems*. [S.l.]: CRC Press, 2003.

[11] GANSSLE, J. et al. *Embedded Hardware: Know It All*. [S.l.]: Newnes, 2007.

[12] LEE, E. A.; SESHIA, S. A. Introduction to embedded systems. *A Cyber-Physical Systems Approach*, v. 2, 2015.

[13] CATSOULIS, J. *First Steps with Embedded Systems*. Second edition. [S.l.]: O'Reilly Media, Inc., 2005.

[14] SINCLAIR, I. *Sensors and transducers*. [S.l.]: Newnes, 2000.

[15] OGATA, K. *Modern Control Engineering*. Quarta edição. University of Minessota: Editora Pearson Hall, 2002. 976 p.

[16] ÅSTRÖM, K. J.; HÄGGLUND, T. *PID controllers: theory, design, and tuning*. [S.l.]: Isa Research Triangle Park, NC, 1995.

[17] VISIOLI, A. *Practical PID control*. [S.l.]: Springer Science & Business Media, 2006.

[18] GUMSTIX. *OVERO FIRE COM. Product Datasheet*. Disponível em: <https://s3-us-west-2.amazonaws.com/media.gumstix.com/datasheets/GUM3503F.pdf>.

[19] GUMSTIX. *OVERO WATER COM. Product Datasheet*. Disponível em: <https://s3-us-west-2.amazonaws.com/media.gumstix.com/datasheets/GUM3503W.pdf>.

[20] INSTRUMENT, T. *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator*. Disponível em: <http://www.ti.com/lit/ds/symlink/lm2596.pdf>.

[21] XLSEMI. *5A 180KHz 36V Buck DC to DC Converter*. Disponível em: <http://i-makers.info/resource/XL4015%20datasheet.pdf>.

[22] ST. *LOW DROP FIXED AND ADJUSTABLE POSITIVE VOLTAGE REGULATORS*. Disponível em: <https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf>.

[23] WILLIAMS, T. *The Circuit Designer's Companion*. Segunda edição. Grã-Bretanha: Editora Elsevier, 2005. 354 p.

[24] ST. *Ultra compact high performance e-compass 3D accelerometer and 3D magnetometer module*. Disponível em: <https://cdn-shop.adafruit.com/datasheets/LSM303DLHC.PDF>.

[25] ST. *MEMS motion sensor: three-axis digital output gyroscope*. Disponível em: <http://www.st.com/content/ccc/resource/technical/document/datasheet/43/37/e3/06/b0/bf/48/bd/DM00036465.pdf/files/DM00036465.pdf/jcr:content/translations/en.DM00036465.pdf>.

[26] BOSCH. *Digital pressure sensor*. Disponível em: <https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>.

[27] KIM, S.-G. *2.008 Design & Manufacturing II, Lecture Notes*. Instituto de Tecnologia de Massachusetts: MIT, 2004. Disponível em: <https://ocw.mit.edu/courses/mechanical-engineering/2-008-design-and-manufacturing-ii-spring-2004/lecture-notes/04_polymer_1_6_f.pdf>.

[28] SCIAVICCO, L.; SICILIANO, B. *Modelling and control of robot manipulators*. [S.l.]: Springer Science & Business Media, 2012.

[29] PAPACHRISTOS, K. A. C.; TZES, A. Design and experimental attitude control of an unmanned tilt–rotor aerial vehicle. *The 15th International Conference on Advanced Robotics*, p. 465–470, jun. 2011.

ANEXOS

I. DIAGRAMA ELETRÔNICO DO MÓDULO DE ALTA CORRENTE

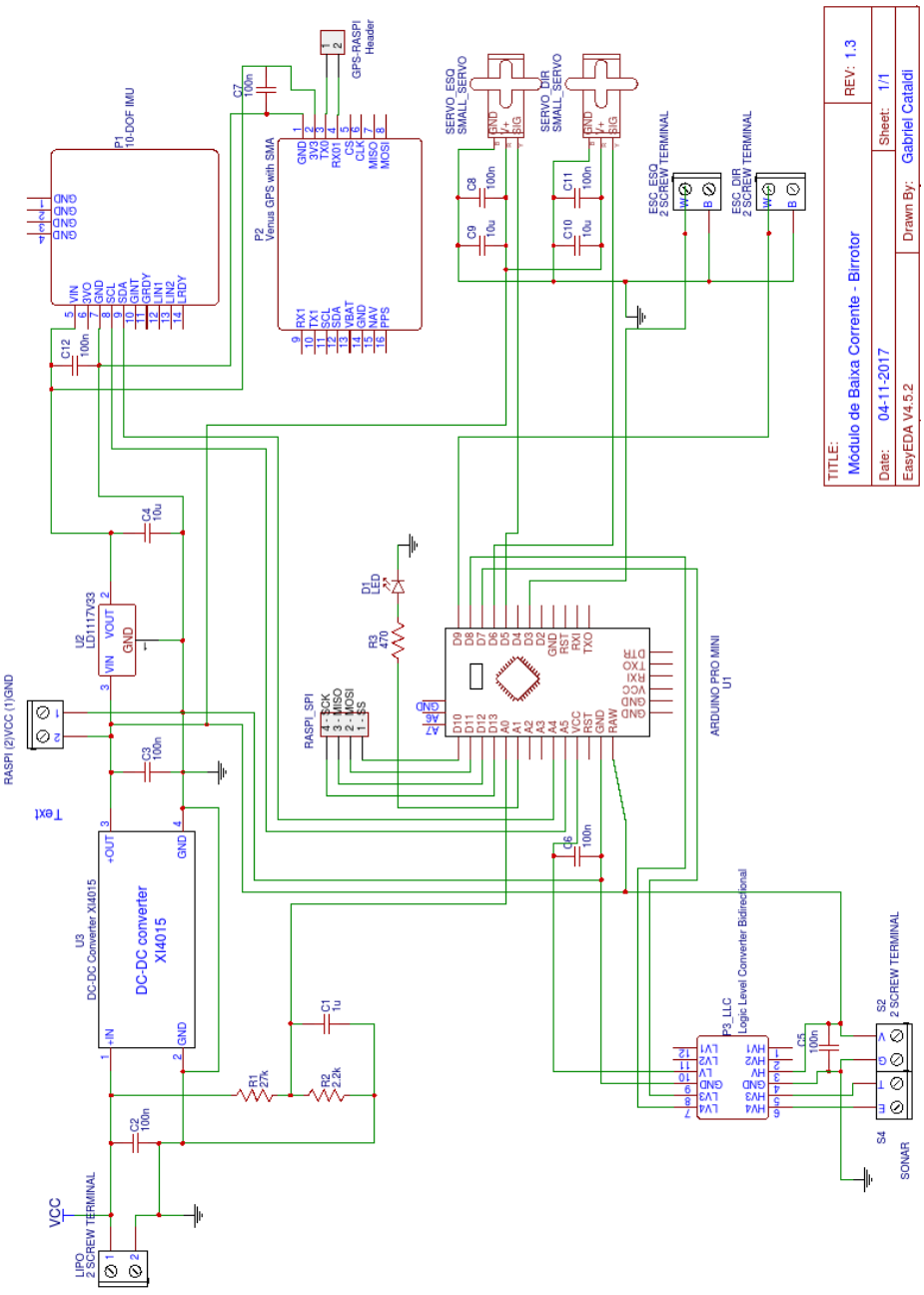


Figura I.1: Versão ampliada do diagrama eletrônico do módulo de alta corrente.

II. DESENHO TÉCNICO

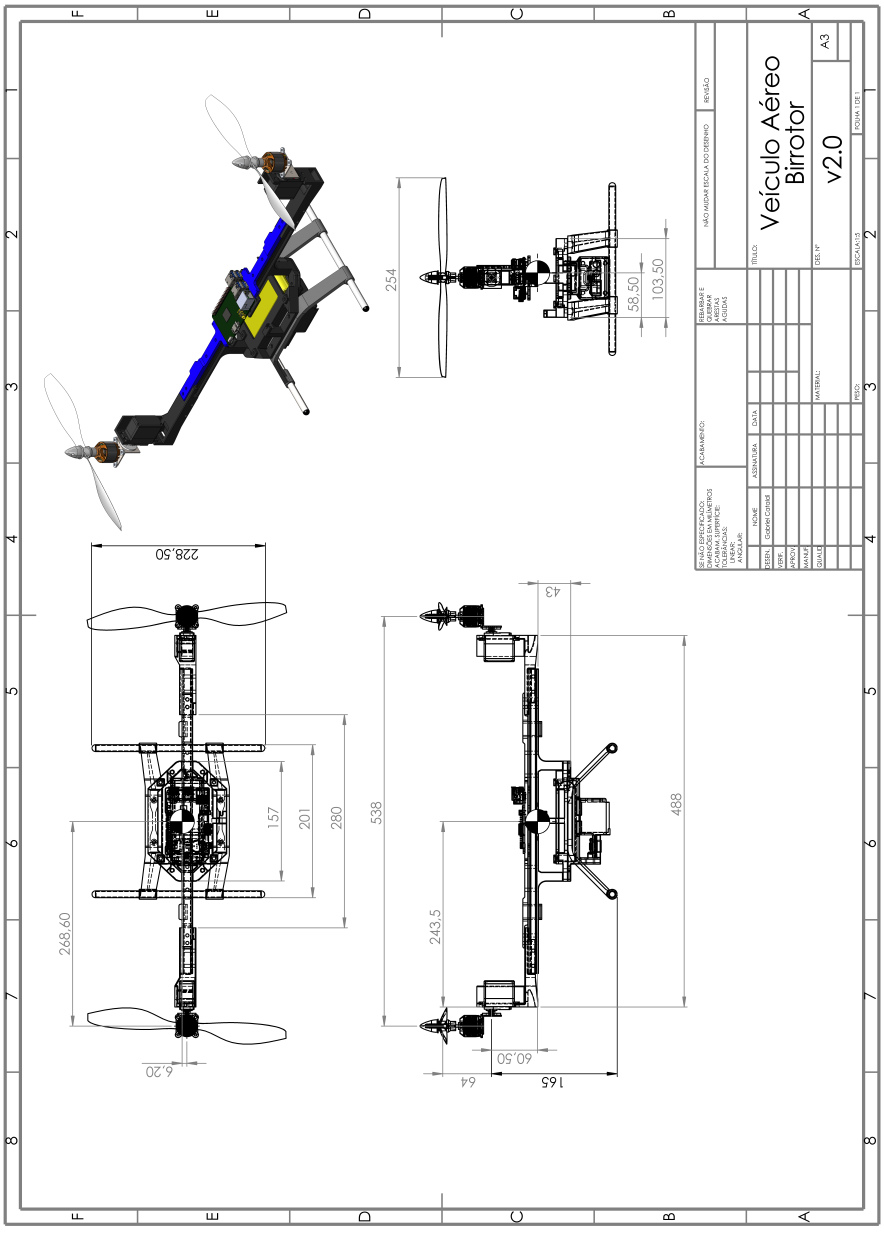


Figura II.1: Desenho técnico do veículo aéreo birrotor.

III. DESCRIÇÃO DO CONTEÚDO DO CD

Em um CD entregue junto com esse documento estão disponibilizados uma cópia digital do relatório, e os projetos mecânico, eletrônico e de *software*, com todos os arquivos utilizados. No sítio https://github.com/gcataldi/Navi estão disponíveis todos os arquivos do *software*.